إدوارد آنجل

رسومات الحاسوب

ترجمة:

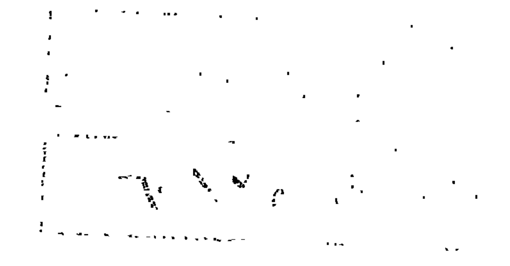
الدكتور نعمة عبدالله السلطاني

الأستاذ الدكتور باقر الهاشمي



اهداءات ٢٠٠٢
دار المناهج للنشر والتوزيع
سلطنة عمان


BIBLIOTHECA ALEXANDRINA
مكتبة الاسكندرية




رسومات الحاسوب
1
الجزء الأول

بسم الله الرحمن الرحيم



هاتف ٤٦٥٠٦٢٤
فاكس (٠٠٩٦٢٦) ٤٦٥٠٦٢٤
ص.ب. ٢١٥٣٠٨
عمان ١١١٢٢
الأردن
الموقع
عمان
شارع الملك حسين
بناية الشركة المتحدة للتأمين

الطبعة الأولى
٢٠٠١م - ١٤٢١هـ

رقم الإجازة المتسلسل لدى دائرة المطبوعات والنشر ١٥٢٥/١٠/٢٠٠٠
رقم الإيداع لدى دائرة المكتبات والوثائق الوطنية ٢٨٢١ /١٠/٢٠٠٠

Converted by Tiff Combine - (no stamps are applied by registered version)

إدوارد آنجل

رسومات الحاسوب

1

الجزء الأول

ترجمة:

الدكتور نعمة عبدالله السلطاني   الأستاذ الدكتور باقر الهاشمي

◆◆◆◆◆◆◆◆◆





المحتويات

الفصل الأول
نظرة عامة

الفصل الثاني
أفكار أساسية

الفصل الثالث
رسومات ثنائية البعد

الفصل الرابع
رسومات متفاعلة

الفصل الخامس
التحويلات والنمذجة

مقدمة المترجمين

لقد وجدنا في هذا الكتاب مادة غنية في موضوع رسومات الحاسوب مع التطبيقات الحديثة وأهمية في مجال الحاسبات وتقنيات البرمجة والتي قدمها المؤلف في مقدمته لهذا الكتاب. وهذا مما دفعنا في ترجمة مثل هذا الكتاب الذي يستفاد منه طلبه علوم الحاسبات وهندسة الحاسبات وغيرهم من الاختصاصات ذات العلاقة.

كما ورد في مقدمة المؤلف إن هذا الكتاب يمكن تقسيمه إلى جزئين أولهما يتضمن الفصول الخمسة الأولى وتعتبر كبداية منهجية لدراسة رسومات الحاسوب، وثانيهما يتضمن الفصول الخمسة الباقية وتعتبر مادة متقدمة في موضوع الرسومات. لهذا تم وضع هذا الكتاب المترجم في جزئين على ما ورد في مقدمة المؤلف.

يسعدنا أن نقدم هذا الجهد المتواضع بين يدي القارئ العربي وخاصة أبنائنا طلبة الجامعات والمتخصصين في هذا المجال ، آملين الاستفادة منه والله من وراء القصد.

المترجمان

المقدمة

Preface

لقد رأت السنين الماضية ثورة في طريقة استخدام الرسومات في الحاسوب ، حيث لا تزال رسومات الحاسوب تفتح آفاق جديدة من الفعاليات في علم الحاسوب. ولهذا المجال أهمية للطلبة المحترفين في الهندسة، العلوم ، الأعمال والرياضيات. أن التقدم الذي حصل في المكونات المادية والبرمجيات معاً أدى إلى اعتبار محطة العمل الحديثة لرسومات الحاسوب كأداة قياسية. إن محطات العمل هذه لا تشتمل فقط على مكونات مادية عالية الدقة وعارضات ذات مخططات الرقم الثنائي (Bit-mapped) ، ولكن تشمل أيضاً على وسائل ضرورية لبرمجيات جاهزة والتي تسمح للمستفيدين القيام بتطوير تطبيقات خاصة بهم.

هكذا نرى المهندس الكهربائي يكتب واجهة بينية إلى حزمة برامج تحليل الدوائر الكهربائية (Circuit-Analysis package) وكذلك المهندس الميكانيكي يكتب حزمة برامج للتصميم المعزز بالحاسوب (Computer- Aided Design) والمدار بقائمة اختيارات (Menu- driven) . وأما بالنسبة لاختصاصي علم الحاسوب، فالتوسع الذي طرأ في مجال رسومات الحاسوب قد شمل ثروه جديدة ومثيرة من التطبيقات ابتداءً من توليد صور واقعية الإضاءة (Photorealistic Displays) إلى تصميم وسائل برمجية جاهزة.

إن الهدف من هذا الكتاب هو استخدامه في تدريس مادة رسومات الحاسوب لطلبة المراحل المتقدمة في علم الحاسوب والهندسة ولأول مرة، حيث تكون المتطلبات الوحيدة لهذه المادة هي مهارات جيدة في البرمجة والرياضيات فقط. ويعتمد هذا الكتاب على أربعة مبادئ أساسية هي:

أولاً: عند توفر الوسائل الحاضرة للبرمجيات والمكونات المادية يكون من الضروري للطلبة أن يبدؤا العمل على تطبيقات مهمة في رسومات الحاسوب في مرحلة مبكرة من تدريس المادة. ولهذا تبنى الكتاب الأسلوب الهرمي (أي من الأعلى إلى الأسفل Top-down) لكي يتسنى للطالب العمل على مشاريع يستخدم فيه منظومه نواة رسومات الحاسوب (Graphical Kernel System-GKS) أو أي منظومه أخرى، وقبل أن يمر وقتاً طويلاً على دراسة خوارزميات

رسومات الحاسوب، كرسم الخط والملء (Filling) . لذلك يتبع هذا الكتاب أسلوب مغاير إلى معظم الكتب التي تعتمد على أسلوب من الأسفل إلى الأعلى (Down-top) حيث يبدأ مع عناصر الصورة (Pixel) ويعمل للأعلى ماراً بالخطوط وأخيراً يصل إلى التطبيقات.

ثانياً: أن تبني منظومة GKS كلغة قياسية لرسومات الحاسوب لها أهمية كبيرة في عملية تدريس الموضوع . بالرغم من نواقصه، توفر منظمومة GKS أسس مفاهيمية لتدريس رسومات الحاسوب الذي يشترك مع العديد من منظومات، حيث تسمح لنا بالتدريس مستخدمين أدوات يمكن نقلها بسهولة إلى منظومات أخرى.

ثالثاً: إن الزيادة المتسارعة في مهارات البرمجة لكل من المهندسين واختصاصي علم الحاسوب، كانت لها أثر مهم على هذا الكتاب. نرى أن المادة المتعلقة بموضوع النمذجة مع هياكل معلومات هرمية، والتي تقوم معظم الكتب بوضعها في نهاية الكتاب أو حذفها كلياً، الآن أصبحت ضمن إمكانية طلبة الصفوف غير المنتهية في علم الحاسوب والهندسة. تعتبر الفصول المتعلقة بالتحويلات والنمذجة الهرمية جزء رئيسي من هذا الكتاب ولهذا ظهرت مبكرة.

رابعاً: إن ضرورة التوسع في معرفتنا لرسومات الحاسوب يتطلب على الأرجح وجود مقررين على الأقل في موضوع رسومات الحاسوب ضمن معظم أقسام علم الحاسوب هما:

1. مقدمة تقوم بتغطية سلسلة من المجالات مع أهمية التأكيد على المنظومات وأوجه هندسة البرمجيات لرسومات الحاسوب.
2. يتبعه موضوعات تؤكد على الخوارزميات، والنمذجة الهندسية واقتفاء أثر الشعاع (Ray Tracing) والهندسة الحسابية. لقد تم تصميم هذا الكتاب كمقدمة لرسومات الحاسوب.

إن اتخاذ القرار في اختيار لغة C ومستوى التفصيل لمنظومة GKS تم بعد تفكير عميق لآراء عدد من المختصين . إن لغة C توفر لنا التوازن بين الرغبة في توفير بعض التجريد في حين نبقى إلى حد ما بالقرب من الماكنة التي يمكن مناقشة قضايا التنفيذ عليها. في الحقيقة هي اللغة المفضلة في الوقت الحاضر من قبل منفذي منظومات رسومات الحاسوب وتعتبر هذه فائدة إضافية أخرى. إن مستوى لغة C المستخدمة هنا لا تسبب أي عائق للطلبة الذين يعرفون فقط لغة باسكال أو فورتران. قد يكون مستوى تفاصيل ترابط لغة C بمنظومة GKS أكثر إثارة للجدل.

وجدنا من خبرتنا مع الكتب الأخرى معظم وقت المحاضرة يهدر في توضيح أو تصحيح مادة سطحية تتعلق ببعض لغات رسومات الحاسوب الخاصة المستخدمة في تلك الكتب.حاولنا

هنا الوصول إلى توازن باستخدام مجموعة جزئية من دوال GKS مع توفير تفاصيل حول هـــذه الدوال المستخدمة قدر الإمكان. وكما نعتقد يكون من الضروري أن نرى التفاصيل حتى إذا لم يكتبوا الطلبة برامج. ولهؤلاء الذين معهم منظومات برمجيات أخرى كمنظومة PHIGS أو بعض المنظومات التجارية، قد يتطلب حد أدنى من التغييرات الضرورية لتحويل برامج GKS.

يحتوي الفصول الثمانية الأولى يضم مادة أساسية للصفوف المتقدمـــة وبصـــورة خاصـــة لتخصص علوم كحاسوب ومهندسوا الحاسوب. بالإمكان تدريس مادة الكتاب علـــى مـــدى فصلين وذلك بالتعمق في تدريس بعض الخوارزميات وإجراء مسح لأدبيات الفصلين الأخـــيرين. وأما بالنسبة للتدريسات التي تم فيها التركيز على موضوع المشروع، فيمكن تدريس مادة فصول الكتاب (من 1 إلى 5). في الفصل الأول ومادة الفصول من (6–10) في الفصل الثاني. تـــدرس الفصول (من 1 إلى 4) ولربما الخامس بالترتيب. وأما الفصول ستة، سبعة، ثمانية، تسعة وعشـــرة نوعا تكون مستقلة عن بعضها البعض. وأخيراً يمكن تداول الكتاب من قبل المبرمجين المحـــترفين والمهندسين واختصاصي علم الحاسوب وغيرهم.

الفصل الأول

نظرة عامة (Overview)



الفصل الأول
نظرة عامــــة (Overview)

1. المقدمة (*Introduction*)

التطورات التي حدثت في المجالات العلمية والتقنية والتجارية والصناعية كانت تعتمد دوماً على قابليتنا في توصيل المعلومات، سواء من خلال إرسالها على شكل بتات (أرقام ثنائية-Bits) مخزونة في رقيقه مايكروية (Microchip) أو من خلال المحاورة الصوتية. يعتبر الإنسان محظوظاً لكونه يمتلك منظومة رؤيا متطورة ومعقدة التركيب.

وبالتأكيد لا زال القول المأثور "الصورة تعادل عشرة آلاف كلمة" قائماً في وقتنا هذا كما كان عليه قبل مائة سنة. هكذا لم يكن من المفاجئ أن ترى حال ظهور الحاسبات محاولة عدد من الباحثين استخدامه لإخراج صور على شاشة أنبوبة الأشعة الكاثودية (Cathod- Ray Tubc: CRT) . ومن خلال التطور المستمر للحاسبات الإلكترونية على مدى أكثر من أربعين عاماً الماضية ازدادت قدراتنا على إنتاج الصور المولدة بالحاسوب (Computer – Generated Images) إلى حد أن توفرت الآن إمكانات الرسوم حتى على الحاسبات البسيطة.

أخذت رسومات الحاسوب (Computer Graphics) تدخل في كثير من المجالات المتنوعة. حيث تراوحت التطبيقات من رسم مخططات وخطوط بيانية (Charts and Graphs) إلى توليد صور واقعية (Realistic Images) للتلفزيون والصور المتحركة إلى التصميم التفاعلي لقطع غيار ميكانيكية. هكذا يمكننا شمول جميع هذه التطبيقات في تبني تعريف بسيط وهو :

بأن رسومات الحاسوب هي كل ما يتعلق بالنواحي التي يُستخدم فيها الحاسوب لتوليد الصور. وحسب هذا التعريف تتضمن رسومات الحاسوب ما يلي:

1- تصميم المكونات المادية (Hardware) كالعارضات الصورية (Displays).

2- الخوارزميات الضرورية لتوليد الخطوط على هذه العارضات.

3- البرمجيات (Software) المستخدمة من قبل كل من مبرمجي منظومة رسومات الحاسوب ومبرمجي التطبيقات.

4- تطبيقات الصور المولدة بالحاسوب.

إن دراستنا لموضوع رسومات الحاسوب يمكن أن يأخذ اتجاهات عديدة، حيث تمتد هذه الدراسة من دراسة المكونات المادية لرسومات الحاسوب إلى دراسة استخدام الرسومات في حقل متخصص فقط، مثلاً ، تصميم الدوائر المتكاملة ذات الكثافة العالية جداً (Very Large Scale Integration – VLSI).

هنا سيكون توجهنا لدراسة الموضوع من وجهة نظر مبرمج التطبيق (Programmer Application) . كلما توفرت مكونات مادية وبرمجيات منطورة وجيدة ، تصبح دراسة رسومات الحاسوب مهمة بالنسبة للمستفيد كأهميته لمصممي منظومة رسومات الحاسوب. لقد أتاحت محطة عمل (Workstation) لرسومات الحاسوب الحديثة القيام بتصميم تطبيقات خاصة به مستخدماً منظومات رسومات متطورة بدلاً من حاجته إلى استدعاء برمجيات جاهزة سحرية! وبالعكس، غالباً ما يكون نفس التوجه مفيداً لمصممي المكونات المادية ومبرمجي المنظومة (System Programmer)، حيث ينطلب منهم تنفيذ دوال قد تنشأ من حاجة التطبيق.

وبصورة عامة، لا يمكننا كتابة برامج تطبيقية لرسومات الحاسوب من غير الاطلاع على الأساليب التنفيذية، خشية من أن نتوقع الكثير من المنظومة وبالحقيقة لا تستطيع إنتاجه. لذا يكون من الضروري أن تكون لدينا فكرة كاملة عن : المكونات المادية، البرمجيات، الخوارزميات والتطبيقات. هكذا سنبدأ بتطوير برامج المستفيد ومن ثم نستخدمها كقاعدة في مناقشتنا للقضايا المتعلقة بالتنفيذ.

1.1 تطبيقات رسومات الحاسوب:*(Applications of Computer Graphics)*

يمكننا تصنيف تطبيقات رسومات الحاسوب إلى أربعة مجالات رئيسية:

1- عرض المعلومات (Display of Information)

2- التصميم (Design)

3- المحاكاة (Simulation)

4- واجهات بينية للمستفيد (User Interfaces)

بالرغم من أن أكثر التطبيقات تمتد إلى مجالين أو ثلاثة أو حتى تشمل جميعها، لكن التطور الذي حصل في هذا التخصص كان يستند على العمل المتواصل في كل مجال على انفراد.

1.1.1 عرض المعلومات: (Display of Information)

الرسومات كانت دوماً مرتبطة مع عرض المعلومات.

أمثلة على ذلك:

1- استخدام الإسقاطات المتعامدة (Orthographic Projections) لغرض مخططات طوابق الأبنية التي وجدت في عصور البابليين التي تعود إلى 4000 سنة (Carl 78).

2- استخدمت طرق ميكانيكية لخلق رسومات منظورية (Perspective Drawings) في عصر النهضة الأوربية.

3- نرى أن عدد لا يحصى من طلبة الهندسة والعلوم يقومون بتحليل نتائجهم المختبرية وغيرها بواسطة رسومات بيانية على المخططات اللوغاريتمية.

4- منذ عهد قريب ظهرت حزم من البرمجيات (Software Packages) تتيح للمستفيد التصميم التفاعلي (Interactive Design) لمخططات بيانية مجسدة باللون ومجموعات بيانية متعددة. حيث أصبحت تبادل طرق الرسم ظاهرة اعتيادية.

5- في اختصاصات عديدة كفن العمارة والتصميم الميكانيكي حلت محل الرسومات الهندسية التي كانت ترسم باليد، منظمات الرسم الهندسي معتمدة على الحاسوب ومستخدمة أجهزة الراسم الخطي (Plotters) ومحطات العمل (Warkstations).

6- التصوير الطبي (Medical Imaging) يستخدم رسومات الحاسوب في عدة تطبيقات مثيرة.

7- ظهرت في الآونة الأخيرة اهتمامات كبيرة في مشكلات الخيال أو الرؤيا العلمية (Scientific Visualization). مع أن الباحثين أخذوا يستخدمون الآن الحواسيب العملاقة (Super Computers) في وضع حلول أساسية لهذه المسائل التي كانت مستعصية سابقاً في مجالات مثل تدفق السوائل (Fluid Flow)

وبايولوجية الجزئيات (Molocular Biology). لكن كل هذه التطبيقات تحتاج إلى تقنيات عرض جديدة من أجل تفسير نتائج التحليل لكميات هائلة من بيانات مولدة متعددة الأبعاد (Multidmesianal Data Generated).

2.1.1 التصميم : (Design)

هنا تكون المهن الهندسية والمعمارية معنية بعملية التصميم. بالرغم من أن تطبيقاتهم متباينة، لكن معظم المصممين يواجهون صعوبات متشابهة حيث يستعينون بحلول منهجية مماثلة. إن إحدى السمات الأساسية لأكثر مشاكل التصميم هي الافتقار لإيجاد حل وحيد. ولهذا السبب يبدأ المصمم بفحص التصميم الأولي ومن ثم يقوم بتعديله، ربما لعدة مرات، محاولاً تحقيق أفضل حل لهذا التصميم. لذا أصبحت رسومات الحاسوب عنصر لا يمكن الاستغناء عنه في عملية التصميم المتكررة هذه.

لنأخذ مثالاً نتأمل فيه كيف تشترك رسومات الحاسوب في عملية تصميم دوائر إلكترونية (انظر إلى الصورة7).

هنا يجلس المصمم أمام محطة عمل للرسومات مزودة بجهاز إدخال بياني (Graphical Input Device) مثل الفأر (Mouse) وبواسطته يمكن الإشارة إلى مواقع مختلفة على العارضة. نفترض في بادئ الأمر قد تحتوي شاشة العرض على مجموعة رموز مختلفة تمثل عناصر الدائرة ومساحة فارغة يتم فيها تركيب الدائرة المطلوبة. يبدأ المصمم استخدام جهاز الإدخال لاختيار وتحريك رموز العناصر المطلوبة في التصميم وعمل التوصيلات فيما بينهما. لذا نرى للحصول على هذا التصميم الابتدائي، تحتاج المنظومة إلى استخدامات متطورة ومعقدة لرسومات الحاسوب. إذن يتم رسم عناصر الدائرة، وربما يتم تحريكها على الشاشة. وذلك باستخدام جهاز الإدخال للإشارة إلى أحد الاختبارات لرموز العناصر ومواقع رسمها. قد يتم استخدام عدد من الوسائل التي تساعد المصمم في تحديد مواقع رموز عناصر الدائرة بدقة وإجراء بعض الوظائف مثلاً، تحديد مسار الأسلاك ذاتياً.

في هذه المرحلة لربما يرغب المصمم اختبار دائرته المصممة. وسيتم اختبار الدائرة بواسطة برنامج تحليلي يقوم بعرض نتائج التحليل (مثال ذلك منحنيات بيانية للفولتيات مقابل الزمن) على شاشة محطة العمل. الآن يستطيع المصمم أن يدخل بعض التعديلات

على التصميم حسب الضرورة أو يختار تصميم آخر أو يقبل إلى ما توصل إليه.نستنتج من هذا المثال، إن المصمم لا يحتاج أن يكتب برامج رسومات ولا حتى معرفة كثيرة عن الرسومات بالحاسوب، ولكن مع ذلك، من غير الرسومات بالحاسوب عملية التصميم قد لا تكون ممكنة.

3.1.1 المحاكاة: (Simulation)

يمكن تصنيف بعض استخدامات رسومات الحاسوب المثيرة للإعجاب والمألوفة تحت المحاكاة (Simulation) . أعطيت ألعاب الفيديو الدليل الواضح للرؤية الجذابة لرسومات الحاسوب وقدرتنا في توليد صور معقدة آنياً في الوقت الحقيقي (Real-time) . إن التفاصيل الداخلية للعبة القناطر أو الأركد (Arcade Game) تكشف لنا آخر التطورات (State-of-art) التي وصلت إليها المكونات المادية والبرمجيات. وكذلك تمثل الصور المولدة بالحاسوب جزء رئيسي من أنظمة محاكي الطيران (Flight Simulators) . (انظر الصورة8) والتي أصبحت الطريقة المعتمدة والقياسية في تدريب الطيارين. لقد تم إدراك مدى المردود الاقتصادي الهائل والمحافظة على الأرواح باستخدام منظومات محاكاة كهذه. إن التقدم الذي حصل في توليد الصور بالحاسوب والتي نشاهدها في التلفزيون أو في الأفلام السينمائية أحياناً لا يمكن تميزها عن الصور الحقيقية.

4.1.1 واجهات بينية للمستفيد : (User Interfaces)

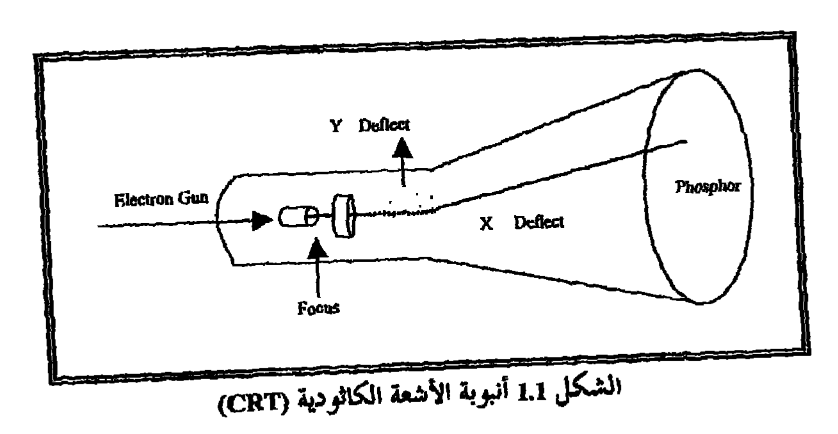
لقد طرأ تغيير جذري على الواجهة البينية (Interface) بين الإنسان والحاسوب بسبب استخدام رسومات الحاسوب. تأمل مكتب سكرتارية إلكترونية (Electronic Office) حيث تجلس سكرتيرة أمام محطة عمل، بدلاً من الجلوس أمام منضدة مجهزة بطابعة. يزود هذا المستفيد بجهاز تأشير (Pointing -- Device) كالفأر يمكنه به الاتصال مع محطة العمل . تظهر على شاشة العارضة عدد من الشواخص أو الدلالات (أيكونات Icons) واختيارات تمثل العمليات المختلفة التي يمكن تأديتها من قبل السكرتيرة. على سبيل المثال ، قد تكون هناك دلالة أو أيكون تمثل صندوق بريد (Mailbox) فعند الإشارة إليه والنقر على مفتاح الفأر (Click On) يؤدي إلى ظهور رسائل بريد إلكترونية (Electronic Mail Message) على الشاشة. وعند اختيار دلالة أو أيكون آخر مثل

سلة الأوراق المهملة (Waste Paper Basket) يسمح للسكرتيرة أن تتخلص من الرسائل البريدية التي لا تحتاجها، في حين عند اختيار أيكون ملفات (File Cabinet) الرسائل والوثائق الأخرى يمكن حفظها. قد تكون نفس الواجهة البينية للرسومات جزء من منظومة تصميم الدائرة الكهربائية التي تم شرحها في البند 2.1.1 أعلاه. وضمن سياق هذا الكتاب، سنرى أن هذه الواجهات البينية تعتبر استخدامات واضحة لرسومات الحاسوب. مع أن من وجهة نظر السكرتيرة التي تستخدم منظومة مكننه المكاتب (Office-Automation) أو مصمم الدوائر الكهربائية، تعتبر رسومات الحاسوب جانب ثانوي بالنسبة للعمل المطلوب أداته. برغم أن هؤلاء المستفيدين لا يكتبوا أبداً برامج رسومات، لكن هناك عدد كبير من المستفيدين يستخدموا الرسومات بالحاسوب.

2.1 تطور رسومات الحاسوب : (The Development of Computer Graphics)

1.2.1 1950 إلى 1960

لقد ظهرت الأمثلة الأولى لرسومات الحاسوب في الأيام المبكرة من عصر الحاسوب الحديث. في أوائل الخمسينات، قام فريق من الباحثين في معهد MIT في الولايات المتحدة الأمريكية باستخدام الحاسوب للتحكم في انحراف الحزمة الإلكترونية لأنبوبة الأشعة الكاثودية أو المهبطية (Cathode-Ray Tube-CRT). يبين الشكل1.1 مخطط بسيط لـ CRT.



الشكل 1.1 أنبوبة الأشعة الكاثودية (CRT)

ينبعث الضوء عند اصطدام الإلكترونات بالطلاء الفسفوري الداخلي للأنبوبة. يتـــم السيطرة على موقع الحزمة بواسطة زوج من ألواح الانحـــراف (Deflection – Plates). تقوم المحولات الرقمية – إلى – التناظرية (Digital – to- Analogue Convertor) بتحويل مخرجات (Output) الحاسوب إلى فولتيات عبر ألواح الانحراف في الاتجـــــاهين(x و y). وبواسطة توجيه حزمة إلكترونية كافية الشدة على الطلاء الفسفوري يجعل ظهور ضـــوء على سطح شاشة الـــ CRT. إذا كانت الفولتية الموجة للحزمة تتغير بمعدل ثابت، ستقوم الحزمة برسم خط مستقيم مرئي للمشاهد. يُعرف هذا الجهاز بالأنبوبـــة الكاثوديـــة ذات المسح العشوائي (Random-Scan) أو راسم الخط (Calligraphic) لأنه يمكـــن تحريـــك الحزمة بصورة مباشرة من موقع لآخر على سطح الشاشة. إذا أطفأت شدة إضاءة الحزمـــة، يمكن تحريك الحزمة إلى موقع جديد بدون ترك أي أثر للرؤية على الشاشة.

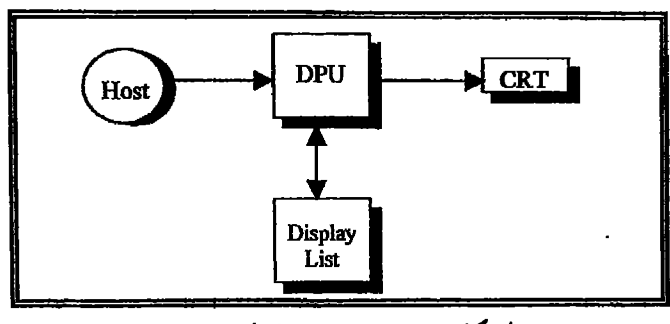
تقوم أنبوبة الأشعة الكاثودية CRT بإطلاق الضوء فقط لفترة زمنية قصيرة جداً وذلـــك بعد إثارة الطلاء الفسفوري بالحزمة الإلكترونية، لكي يرى الإنسان صورة مســـتقرة علـــى معظم عارضات CRT، يتطلب من الحزمة تتبع نفس المسار على الأقل 50 مـــرة بالثانيـــة. كانت الحواسيب الرقمية السابقة غير قادرة على حساب الجهود الكهربائيـــة الضروريـــة بسرعة كافية لتوليد صور مركبة (Complex Picture) . إن هذا القصور، مضافـــاً إليـــه الكلفة العالية للحواسيب السابقة، كان في بادئ الأمر معوقـــاً أمـــام تطـــور رســـومات الحاسوب، ولكن التطورات اللاحقة جعلت CRT جهاز إخراج قياسي لمعظم المنظومات.

2.2.1 1960 إلى 1970 :

لقد حدثت تغييرات مفاجئة خلال الستينات بسبب عدد من الأحداث والتطـــورات في مجال الرسومات. حيث أصبحت أنبوبة التخزين للرؤية المباشـــرة (Direct – Views Starage Tube-DVST) جهاز إخراج نموذجي واطئ الكلفة. بـــالرغم مـــن أن جـــهاز DVST هو جهاز تخزين غير دائمي، لكنه قادر على الاحتفاظ بالصورة علـــى الشاشـــة لساعات، وهكذا يكون من غير الضروري أن يقوم الحاسوب بتحديث العارضة بصـــورة مستمرة. لقد تم إدخال هذا الجهاز في تصميم وحـــدات طرفيـــة (Terminals) حيـــث استخدمت سلسلة من الرموز الخاصة (Special Character) لتنشيط قدراته في الرسم. إن الكلفة المنخفضة نسبياً لهذا الجهاز إضافة إلى استخدامه كجهاز طرفي متنقل قـــد فســـح

المجال في تطوير برمجيات أساسية للرسومات مثل PLOT 10 والقابل للتنقل (Transported) من منظومة إلى أخرى.

إحدى صفات هذا الجهاز (DVST) إنه جهاز لا يكون ملائماً لتطبيقات الزمن الحقيقي (Real – Time) . إن عمله يكون مشابه إلى رسم خط على قطعة ورقة ويبقى الخط هناك إلى حين رفعه، هكذا تكون جميع مخرجات DVST، حيث تبقى على الشاشة لحين مسح الشاشة كاملة. مع هذا الجهاز لا يمكن تغيير أجزاء من الصورة وتحريكها (Animation) حيث لا يسمح الجهاز بالمسح الانتقائي (Selective Eraser) للصورة. إن تطوير معالج للعرض (Display Processor) فسح المجال في توليد رسومات في الزمن الحقيقي (Real-Time Graphics) ويمكن التفاعل معها على CRT ذات المسح العشوائي. إن معالج العرض عبارة عن حاسوب لأغراض خاصة (Special-Purpose Computer) مع مجموعة محدودة من الإيعازات (Instructions) قادر على تنفيذها بسرعة. من بين مهامه الرئيسية، القيام بإنعاش أو تنشيط (Refresh) الـ CRT بمعدل يجعل العارضة تظهر خالية من الارتعاش أو الارتجاج (Flicker-Free). يتم وضع الكيانات الأساسية للرسومات (Graphical Entities) على حاسوب حاضن (Host Computer) في ذاكرة خاصة تدعى ذاكرة العرض (Display Memory) أو ملف العرض (Display File)، التي يمكن الدخول عليها من قبل وحدة معالجة العرض (Display Processing Unit) كما هو مبين في الشكل 2.1.



الشكل 2.1 معمارية معالج العرض

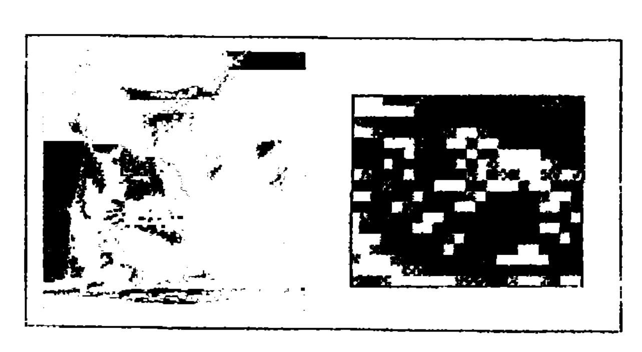
هكذا إذن يحتاج الحاسوب الحاضن تعريف الكيانات الأولية للرسومات (Graphical Primitives) مرة واحدة فقط. حالما ترسل هذه الكيانات الأولية إلى معالج

العرض يتفرغ الحاسوب الحاضن لأداء وظائف أخرى. لقد تم إدمـــاج وظـــائف معـــالج العرض في معظم منظومات الرسومات الحاضرة، وبالرغم من التقدم الحاصل في التقنيـــات قد سمحت لوظائف المنظومات السابقة بأن تختزل في رقيقـــة أو رقيقتـــين (Chips) مـــن الدوائر الإلكترونية المتكاملة.

إن معالج العرض جعل تأدية وظائف أخرى ممكنة، مثلاً، تفاعل المستفيد مع الصـــور المعروضة. حيث بدأ الباحثون تجاربهم على مشروع "Sketchpad" (Suth 62) لإثبـــات مدى إمكانيات هذه المنظومة وذلك من خلال التفحص الدقيق للأهداف المقررة. هكذا تم تشخيص أهمية تراكيب البيانات (Data Structure) والحاجة إلى تطويـــر الخوارزميـــات (Algorithms) واستنباط نماذج تتميز بها رسومات الحاسوب الحديثة.

3.2.1 1970 إلى 1980:

تميزت هذه الحقبة الزمنية (السبعينيات) من تاريخ رسومات الحاســـوب في مجـــيء الرسومات الشبكية (Raster Graphics) . وبسبب الإنخفاض الذي حصل في كلفة إنتاج الذاكرة ذات الحالة الصلبة (Solid – State Memory)، أصبح مجدياً بنـــاء منظومـــات تستخدم فيها CRT ذات المسح الشـــبكي (Raster-Scan CRT). تخزيـــن الصـــور في الرسومات الشبكية على شكل مصفوفة تحتوي على عناصر الصورة (Picture Elements) والتي تدعى باختصار (Pixels) (انظر الشكل 3.1).



الشكل 3.1 عناصر الصورة (Pixels)

بدلاً من مجموعة قطع خطوط (خطوط مستقيمة) كما هو الحـــال في رســـومات المسح العشوائي. يتم خزن عناصر الصورة في مساحة خاصة من الذاكرة تدعى المخـــزن

الانتقالي للصورة (Frame Buffer). تقوم المكونات المادية للعارضة بمسح المخزن الانتقالي للصورة عادة بمعدل 50 إلى 70 مسحة بالثانية، منعشاً بذلك العارضة خط بعد خـــط، وهذا يشبه إلى حد كبير الطريقة المتبعة في إنتاج الصور التلفزيونية التجارية. يتم عـــرض كيانات الإخراج للرســـومات (Graphical Output Primitives)، كقطــع الخطـــوط والنصوص بإضاءة أو إطفاء عناصر الصورة في المخزن الانتقالي للصورة لتكوين الشـــكل التقريبي لهذه الكيانات. إن هذه العملية تعرف بالتحويل المسحي (Scan-Conversion) أو التشابك (Rasterization). في هذه الفترة أي السبعينيات تم تطوير عـــدد كبـــير مـــن خوارزميات التحويل المسحي.

إن القدرة في بناء حسابات صغيرة وسريعة وذات كلفة منخفضة أدت إلى ظـــهور حاسبات شخصية سبقت محطات العمل الحديثة. في معمارية هذه الحاسبات، كان المخـزن الانتقالي للصورة جزء من الذاكرة الرئيسية للحاسوب، لذا بدلاً من أن تكـــون عارضـــة الرسومات جهاز ملحق أصبح جزء مكمل في الحاسوب. وكذلك وجود القدرة في بنـــاء شبكات (Networks) من هذه الحاسبات فتحت لنا آفاق جديدة وعلى حـــد ســـواء في تطوير المكونات المادية والبرمجيات. هكذا بدأت تنتشر بصورة واســـعة منظومـــات ذات أغراض خاصة (Special – Purpose) للتصميم المعزز بالحاســـوب (Computer aided design-CAD) . وكذلك ظهرت طرق جديدة للتفاعل أو الحوار مع الحاسوب، مثـــل، تطوير واجهات بينية للحوار مسيرة بقائمـــة اختيـــارات (Menu-Driven Interfaces) مسيطر عليها بواسطة الفأر (Mouse- Controlled).

بسبب الانتشار الواسع في استخدام رسومات الحاسوب أدى إلى اهتمامات جديـــرة بالاعتبار في تطوير لغة برمجة قياسية للرسومات. أدى هذا إلى تشكيل لجنتين كلاهما نشروا تقاريرهم حول اللغة القياسية في عام 1977 وأيضاً ظهرت نسخ من التقارير المنقحـــة في عام 1979. أنتجت اللجنة الأوربية منظومـــة نـــواة الرســـومات (Graphical Kernel System-GKS) وأما اللجنة الثانية في الولايات المتحدة قامت بتطوير منظومـــة CORE. جاء إكمال هذه المشاريع عندما تبنت منظمة التقييس الدولية (International Standards Organization ISO) منظومة GKS في عام 1984. وبعد فترة قصيرة تبـــنى المعـــهد

القومي الأمريكي للتقييس (ANSI) رسمياً منظومة GKS. سوف نستخدم من هنا فصاعداً منظومة GKS في أمثلتنا القادمة في البرمجة.

4.2.1 من عام 1980 إلى وقتنا الحاضر:

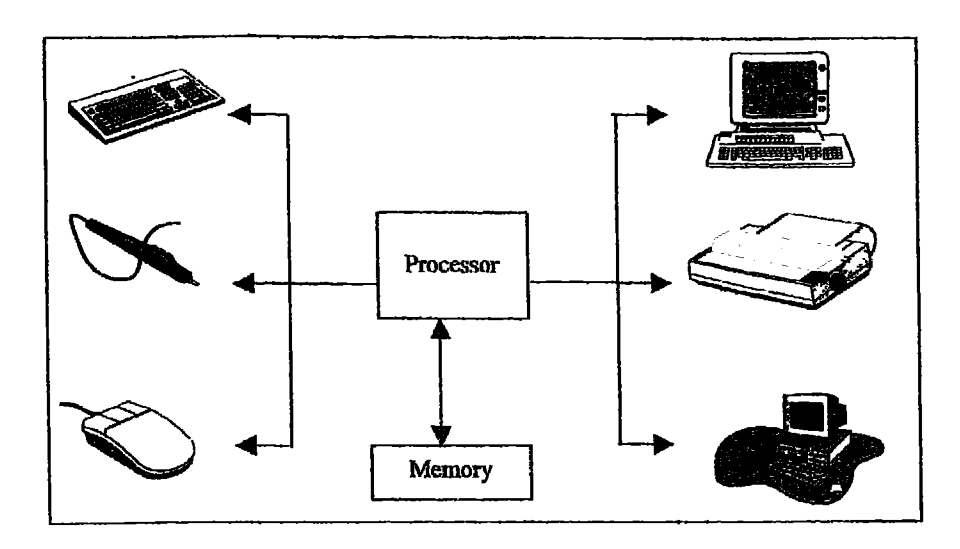
خلال الثمانينات تغير التوجه بعض الشيء. لقد أصبحت محطات العمل الفعالة للرسومات أداة قياسية يستخدمها المهندسين والعلماء وفنانو الرسم والمعماريون. إن مثل محطات العمل هذه، ليس فقط وفرت قوة حسابية هائلة وضعته في متناول يد المستفيد فحسب بل أيضاً زودته بمجموعة وسائل للرسومات على الحاسوب- مثال ذلك، مكتبات جاهزة تحتوي على برامج فرعية للرسومات (Libraries of Graphical Routines) وواجهات بينية مرئية سهلة الاستخدام وكذلك الوصول إلى إمكانات مختلفة من خلال شبكات الحاسوب. محطات العمل التي تستخدم في الخيال أو لرؤيا العلمية (Scientific Visualization) يمكنها إنتاج صور من المحاكاة عند سرع قريبة إلى ما كانت تصل إليه الحواسيب العملاقة (Super Computers) سابقاً.

الآن أصبح بمقدور مستخدمي منظومات CAD أن يعملوا مع منظومات تفاعلية لتصميم نماذج مجسمة (Solid Modelers). وكذلك أصبح بإمكان صانعوا الرسومات المتحركة (Animators) والرسامون التجاريون إنتاج صور واقعية الإضاءة (Photorealistic Images) وذلك باستخدام المكونات المادية والبرمجيات المتوفرة تجارياً.

بالرغم من أن التقدم مستمر في تطوير الخوارزميات والبرمجيات القياسية وقدرات المكونات المادية وفي عدة مجالات أخرى تتعلق بالرسومات، أيضاً يوجد هناك قبول عام للأساسيات النظرية والعملية في هذا الحقل. إننا نتوقع تطورات جديدة ومثيرة مستقبلاً، لكن سنواصل تطبيقاتنا مع الثقة بأن الوسائل الأساسية قد ثبتت لدينا بشكل جيد وستستمر لكي تكون مناسبة في المستقبل.

3.1 منظومة رسومات الحاسوب الأساسية : *(Basic Graphics System)*

لنلقي نظرة على هيكل تنظيمي نموذجي لمنظومة رسومات الحاسوب قد تستخدمه. سيكون تركيزنا في بادئ الأمر على كيفية رؤية مبرمج التطبيقات للمنظومة ولهذا سنهمل التفاصيل المتعلقة بالمكونات المادية الشكل 4.1 يبين مخطط كتلي لمنظومتنا هذه.



الشكل 4.1 منظومة رسومات

تتكون المنظومة من أربعة أجزاء رئيسية:

1- المعالج Processor

2- الذاكرة Memory

3- أجهزة الإخراج Output Devices

4- أجهزة الإدخال Input Devices

إن هذا النموذج إلى حد ما عام يشمل محطات العمل، الحواسيب الشخصية، طرفيات مربوطة بحاسوب مركزي ذات المشاركة الزمنية (Time-Shared) ومنظومات متطورة لتوليد الصور (Sophisticated Image- Generation) . لاحظ في كثير من النواحي يمثل هذا المخطط الكلي الحاسوب الشخصي: إن كيفية تخصص كل عنصر لرسومات الحاسوب هو الذي تميز به هذا المخطط كونه منظومة رسومات حاسوب بدلاً من أن يكون حاسوب للأغراض العاملة .

1.3.1 المعالج: (Processor)

ضمن مربع المعالج المبين في الشكل 4.1 قد تجري نوعين من المعالجة:

أولاً: معالجة تكوين الصورة (Picture Formation Processing) في هذه المرحلة يتم معالجة برنامج المستفيد (User Program) أو الأوامر (Commands). يتم تركيب الصورة من عناصرها الأولية (مثل الخطوط والنصوص) المتوفرة لدى المنظومة مستخدمين الصفات (Attributes) المطلوبة مثال ذلك لون الخط وطاقم حروف النص (Font). تعتبر الواجهة البينية للمستفيد جزءاً من هذه المعالجة.

نستطيع تحديد مواصفات الصورة بعدة طرق، فمثلاً، من خلال برنامج تفاعلي للطلاء مدار بقائمة اختيارات (Interactive Menu-Driven Painting Program) أو بواسطة برنامج مكتوب بلغة C مستخدماً مكتبة رسومات جاهزة (Graphics Library). لهذه المرحلة يستخدم المعالج الحقيقي وغالباً ما يكون معالج محطة عمل أو حاسبة حاضنة (Host Computer).

ثانياً: المعالجة الثانية تتعلق بعرض الصور (Display Of The Picture). في المنظومة الشبكية (Raster System)، يجب إجراء تحويل مسحي (Scan Converted) لكيانات أولية (Primitives) معينة وإنعاش الشاشة لتجنب إرتجاج الصورة (Flicker). قد نحتاج إلى مدخلات من المستفيد لإعادة تحديد موقع أشياء منظورة (Objects) على العارضة. إن نوع المعالج المناسب لمثل هذه الأعمال قد لا يكون من النوع القياسي الموجود في معظم الحواسيب. غالباً ما تستخدم ألواح إلكترونية (Board) ورقائق لدوائر متكاملة خاصة.

كما لاحظنا سابقاً، إن أحد العناصر التي تتميز بها منظومات الرسومات هو استخدمها لمعالجات العرض. بما أن شرحنا الآن سيبقى عند مستوى الشكل الكتلي للمنظومة، سوف لا نتوسع في شرح تفاصيل هذه المعماريات إلا فيما بعد.

2.3.1 الذاكرة : (Memory)

غالباً ما يستخدم نوعين مختلفين من الذاكرة في منظومات الرسومات وهما:

1- تكون الذاكرة بالنسبة لمعالجة برنامج المستفيد مشابه إلى ما هو موجود في الحاسوب الاعتيادي، لكون يتم تركيب الصورة بواسطة نوع من العمليات الحسابية الاعتيادية (Standard Type of Arithmetic processing).

2- أما من الناحية الثانية، معالجة العرض، تحتاج إلى ذاكرة عرض ذات سرعة عالية حيث يمكن الوصول إليها من قبل معالج العرض. وفي حالة منظومات المسح الشبكي، تكون هذه الذاكرة هي مخزن انتقالي للصورة (Frame Buffer).

عادة تختلف ذاكرة العرض في كل من خواصها المادية وتنظيمها عن ما يكون مستخدماً في معالج الصورة. عند هذه المرحلة، لا نحتاج التعمق في دراسة التنظيم الهيكلي للذاكرة. لكن عليك أن تدرك أن طريقة التنظيم الداخلي للمعالج والذاكرة هي التي تميز المنظومة البطيئة من منظومة توليد الصور في الزمن الحقيقي (الآني) كمنظومة محاكي الطيران. لكن من وجهة نظرنا الحالية، سنحاول التأكيد على أن جميع هذه المنظومات باستطاعتها تنفيذ نفس الوظائف للحصول على نتائج أو مخرجات (Outputs).

3.3.1 أجهزة الإخراج (Output Devices):

تحتوي منظومتنا الأساسية على جهاز إخراج واحد أو أكثر. بما أن العارضات الشبكية (Raster Display) هي الأكثر انتشاراً لذا سنفترض أن منظومتنا تحتوي على جهاز عرض من نوع CRT ذات المسح الشبكي. وكذلك سنعتبر المخزن الانتقالي للصورة هو جزء من ذاكرة العرض. في المنظومة التامة في ذاتها (Self-Contained)، كمحطة عمل، تكون العارضة جزء مكمل من المنظومة، لذا يتم نقل المعلومات بين المعالج والعارضة بصورة سريعة. عندما تكون العارضة جزءاً منفصلاً، كما هو الحال مع طرفية الرسومات (Graphics Terminal)، نلاحظ سرعة الربط أو نقل المعلومات أكثر بطأ. عادة الطرفيات مع عارضات شبكية يجب أن تحتوي على المخازن الانتقالية للصور الخاصة بها، لكي تكون قادرة على إنعاش عارضاتها موقعياً. في منظوماتنا البسيطة هذه، قد تكون لدينا أيضاً عارضات من نوع آخر، مثل جهاز الراسم البياني (Plotter) الذي يسمح لنا بالحصول على نسخة مطبوعة (Hardcopy) من الصورة.

4.3.1 أجهزة الإدخال: (Input Devices)

قد تحتوي منظومة بسيطة على لوحة مفاتيح (Keyboard) فقط لإدخال ما هو ضروري من المعلومات. تقوم لوحة المفاتيح بتجهيز شفرات رقمية مطابقة لسلسلة من ضربات المستفيد على المفاتيح. عادة تفسر هذه السلسلة من الضربات بشفرات تمثل

الرموز (Character Codes). إذا تم تفسير ضربات المفاتيح بصورة منفـــردة أو جماعيـــة كمدخلات رسومات، يمكننا استخدام لوحة المفاتيح كجهاز إدخال مركب (Complex Input Device).على سبيل المثال، يمكن استخدام المفاتيح ذات الأسـهم (Arrow Keys) المتوفرة على معظم ألواح المفاتيح في توجيه حركة المنزلقة (Cursor) على الشاشة.

معظم منظومات الرسومات توفر على الأقل جهاز إدخال آخر. إن الأكثر شيوعاً في الاستخدام هو الفأر (Mouse) والقلم الضوئي (Light-Pen) وعصا التحكـم (Joystick) ولوحة البيانات (Data Tablet). كل واحد من هذه الأجهزة يمكنها أن تزود المنظومـــة بمعلومة موقعية (Positional Information) وكل جهاز عادة يكون مجهز بزر واحـــد أو عدد من الأزرار لتزويد المعالج بالإشارات.

إن من وجهة نظر المبرمج، يوجد هناك عدد هائل من القضايا المهمة الـــتي تتعلـق بأجهزة الإدخال والإخراج منها:

1. ضرورة الأخذ بنظر الاعتبار، كيفية اتصال البرنامج مع هذه الأجهزة .
2. علينا أن نقرر ما هو نوع المدخلات والمخرجات التي يمكن الحصول عليها.
3. ستكون أيضاً لنا اهتمامات في كيفية التحكم بأجهزة متعددة، بحيث نســتطيع اختيار جهاز معين للمدخلات وتوجيه المخرجات إلى مجموعة أجهزة الإخـراج المتوفرة لدينا.

4.1 برمجيات الرسومات: (*Graphics Software*)

نظراً لكون تركيزنا كان منذ البداية موجهاً نحو كتابة برامج رسومات، لذا ســوف نتطرق إلى عدد من القضايا المتعلقة بالبرمجيـــات. كيـــف يســتطيع مســتفيد التطبيــق (Application User) تداول برامجيات في منظومة الرسومات ؟ ما هي المهمات التي ينبغي أن تقوم بها البرمجيات ؟ سوف نتأمل هذه الأسئلة مبدئياً من وجهة نظر مبرمج التطبيقـــات (Application Programmer) الذي قد يريد كتابة برنـــامج راســم بيـــاني (Plotting Program) أو تصميم حزمة برامـــج لتخطيـــط دائـــرة كهربائيـــة (Circuit Layout Package). إن منظومة الرسومات قد تأخذ أشكال تكوينية (Configurations) مختلفـــة. ولكل شكل تكويني أثر على نوع البرمجيات التي يمكن تطويرها وكيف يتم كتابـــة هـــذه

البرمجيات. إن الأنواع الثلاث الرئيسية لهذه المنظومات، وجميعها تحتوي على الأجزاء الـتي تم وصفها في البند السابق، هي:

1. محطات طرفية مربوطة بحاسوب حاضن أو مضيف.
2. وحدات تامة في ذاتها، كالحواسيب الشخصية ومحطات العمل.
3. منظومات للأغراض الخاصة فائقة السرعة، مثل محاكيات الطيران.

1.4.1 برمجيات طرفية القاعدة (Terminal – Based Software):

إلى عهد قريب كانت معظم رسومات الحاسوب تنتج باستخدام طرفيات رسـومات (Graphics Terminals) متصلة بحاسوب حاضن. لعل الميزة الأكثر أهمية لمثل هذا الشـكل التكويني هو المسار البطيء لنقل المعلومات بين الحاسوب الحاضن والعارضـة. إن أجـهزة الإخراج البيانية البطيئة، كمسجلات الأفلام والراسمات الخطية لها نفس الخـواص. عنـد استخدام مثل هذه المنظومات، لا يمكن تكوين أو تغيير الصورة بصورة سريعة، لذلك تكـون قدرتنا في إجراء رسومات في الزمن الحقيقي محدودة. أيضاً تكون عمليات الرسم الشـبكي مقيدة بالزمن المستغرق في إرسال مجموعات كبيرة من عناصر الصورة إلى العارضة. لا يـزال في الإمكان استلام مدخلات من معظم الأجهزة، لأن هذه المدخلات ليست كثيفة جـداً. على سبيل المثال، قد يكون إحداثيات موقع تم استلامه من لوحة بيانات.

في أغلب الأحيان، قد تكون برمجيات المستفيد لهذه المنظومات في هيئة مكتبـة دوال (Library Functions) أو إجراءات (Procedures) تستطيع استحضار القدرات الأساسية لأجهزة الإدخال والإخراج. على سبيل المثال، توجد هناك إجراءات لرسم خطوط ودوائر وقراءة موقع الفأر ومسح (تفريغ Clear) الشاشة. هكذا يقوم المسـتفيد ببنـاء برامـج رسومات وذلك باستحضار إجراءات من المكتبة، لنأخذ مثلاً برنامج C يحتـوي علـى العبارة التالية:

Plot (x – array, y-array, n);

يستطيع هذا الإجراء رسم خط بياني من مصفوفتين للبيانات y,x بعديهما 'n'. يتـم تكوين وتقييس (Scaled) هذه المصفوفات في جزء من برنامج المسـتفيد مسـبقاً قبـل استدعاء هذا الإجراء أو النهج (Plot). وعند استدعاء الإجراء:

Mouse (x,y);

يعيد لك موقع الفأر (إحداثيات y,x)، الذي يمكن بعدئذ استخدامه من قبل البرنامج. إن برنامج المستفيد يرى الرسومات من خلال هذه الإجراءات فقط. لهذا يستطيع المبرمج إما أن يستخدم هذه الدوال المكتبية في برنامجه الرئيسي أو يبني برامج عند مستوى أعلى، أو يكتب برامج فرعية تكون مبنية على أساس ما موجود في مكتبة الرسومات.

على سبيل المثال، في الفصل الثالث، سنقوم بتطوير إجراء لراسم بياني ذاتي التدرج (Self-Scaling) الذي يأخذ بيانات المستفيد وعلامات المحور والعنوان لغرض رسم خط بياني لـ y, x. يتم تنشيط هذا الإجراء باستدعاء المستفيد لهذا الإجراء مرة واحدة، وحيث أنه موجود مع الدوال عند المستوى الأدنى في مكتبة الرسومات.

2.4.1 برمجيات جاهزة : (Turnkey Software)

في الوقت الحاضر، تكون أكثر برمجيات الرسومات جاهزة بصورة كاملة. لذا لا يحتاج المستفيد كتابة برنامج. بدلاً من ذلك يقوم المستفيد بالتفاعل أو التحاور مع العارضة التي تحتوي على قوائم اختيارات (Menus) ودلالات أو شواخص (Icons) ومساحات للعمل إضافة إلى معلومات أخرى. حيث يقوم المستفيد بتركيب رسومات على العارضة مستخدماً جهاز إدخال كالفأر للاتصال مع المنظومة. إن مثل هذه الواجهة البينية لها فوائد مهمة مقارنة مع الطرق التقليدية، مثل كتابة البرامج. تقريباً تكون أكثر استخدامات الرسومات بالحاسوب على محطات العمل وعلى وحدات تامة في ذاتها. بسبب الحاجة إلى تغير الصورة المعروضة باستمرار، هذا النوع من الرسومات يكون غير ممكن على أشكال تكوينية-طرفية القاعدة (Terminal-Based Configurations). في منظومات متخصصة عالية السرعة، مثل ألعاب الفيديو ومحاكيات الطيران تكون الواجهة البينية للمستفيد مع المنظومة متشابهة. حيث تحتوي على تفاعل بين المستفيد وعارضة الرسومات، بدلاً من أن يكون البرنامج المعالج مكتوب من قبل المستفيد. ولكن من وجهة نظرنا، علينا أن نميز بين من يكتب برنامج الرسومات التفاعلية والذي يكون جاهز للاستخدام من قبل الآخرين. لكتابة مثل هذا البرنامج، يتطلب استخدام وسائل منها مكتبة الرسومات. في مثل هذه المنظومات تشمل مكتبات الرسومات على إجراءات لتناول المعالم والسمات الخاصة لهذه المنظومات.

5.1 محتوى الكتاب:

1- في الفصل القادم (الفصل الثاني) سنبحث المفاهيم الأساسية التي تتضمنها رسومات الحاسوب الحديثة. وبعدئذ سوف نلقي نظرة على الرسومات كما يراها مبرمجي التطبيقات والمستفيدين.

2- أما في الفصلين الثالث والرابع سنتعرف على قواعد كتابة برامج ثنائية البعد (Two-Dimensional) أولاً مع مخرجات فقط ومن ثم مع التفاعل. سوف نستخدم في أمثلتنا منظومة نواة الرسومات (GKS) ، ولكن سنؤكد على أوجه التشابه بين منظومة GKS ومنظومات الرسومات الحديثة الأخرى. عند مناقشة كتابة برامج تطبيقية باستخدام منظومات مستقلة عن الجهاز عالية المستوى (High-Level Device Independent)، عادة سوف لا نميز بين كاتب برنامج التطبيق والمستفيد من البرنامج. عند دراستنا لرسومات الحاسوب، غالباً ما يكون الاثنان نفس الشخص.

3- سيختص الفصل الخامس بالتركيز على التحويلات (Transformations) إن هذه التحويلات ليس فقط توفر أسس التطبيقات عالية المستوى (High-Level Applications) المتضمنة رسومات متفاعلة، بل أيضاً تكون حاسمة في تنفيذ منظومات رسومات . أيضاً سنقوم بمناقشة نماذج هرمية (Hierachical Models) لتمثيل العلاقات في التطبيق وبهذه الواسطة سنتعرف على منظومة PHIGS (Programmer's Hierarchical Interactive Graphics Standard) "هرميات المبرمج القياسي للرسومات المتفاعلة ".

4- في الفصل السادس، سنتحول إلى موضوع تنفيذ منظومة الرسومات. ستكون طريقة دراستنا هي تتبع كيان أولي ابتدأ من تعريفه ضمن برنامج التطبيق ولحد عرضه على جهاز مادي (جهاز عرض CRT). سنقوم بدراسة تلك العمليات التي تم ذكرها في الفصول السابقة والتي كانت واضحة لكل من المستفيد وكاتب برنامج التطبيق. وأيضاً سنقوم بمناقشة كيفية عمل أجهزة المكونات المادية البسيطة، وكيف يتم الاتصال بيننا وبين البرمجيات.

5- أما دراستنا لمنظومات المسح الشبكي في الفصل السابع فتشمل قضايا متعلقة بالتنفيذ والتطبيق. إذا أردنا استخدام هذه المنظومات بصورة كاملة، على الأقل في وقتنا الحاضر، لا يمكن استبعاد بعض الخواص المعينة للأجهزة من كتابة البرامج التطبيقية. وكذلك سنقوم بدراسة بعض الجوانب المتعلقة بالرؤية البشرية وبصورة خاصة إدراك الألوان، حيث سنعمل على استخدام أفضل الألوان المتوفرة على معظم محطات العمل الشبكية.

6- يستعرض الفصل الثامن رسومات ثلاثية الأبعاد معتمداً على التوسع في نتائج الفصول السابقة. سيكون تأكيدنا على كتابة برامج تطبيقية ثلاثية الأبعاد مستخدمين برمجيات مستقلة عن الجهاز. ومن أجل كتابة مثل هذه البرامج ، ينبغي علينا أولاً تطوير إمكانيات نموذج الرؤية ثلاثي الأبعاد وقدرة التحويلات ثلاثية الأبعاد. أيضاً سندرس العلاقة بين الطرق التقليدية للرؤية ورسومات الحاسوب.

7- أما مادة الفصلين الأخيرين فهي تتعلق بالتعرف إلى موضوعات متقدمة لرسومات الحاسوب. سنبحث في كيفية توسيع الواقعية في الرسومات المنتجة والنماذج التطبيقية لها. يشرح الفصل التاسع بصورة رئيسية المضلعات (Polygons)، وكيفية استخدامها في إنتاج مشاهد واقعية (Realistic Scenes) مقربه والتي تحتاج إلى حذف السطوح المخفية (Hidden-Surface Removal) ومن ثم طلائها (Rendering). وأما الفصل العاشر فيبحث عن ماذا يحدث إذا استبعدنا الكيانات الخطية مثل الخطوط والمستويات وعملنا مع المنحنيات والسطوح.

بالرغم من استخدامنا لغة C في جميع البرامج، لكن حاولنا أن نجعل هذه الأمثلة واضحة وبسيطة قدر الإمكان، حتى لو دعت الحاجة في كتابة عبارات برمجة أكثر كفاءة. استعراض تفاصيل لغة C الملازمة لمنظومة (GKS) والتي تشتمل على هياكل مهمة تجدها في الملحقين أ و ب.

تمارين

1.1 حاول تحديد مشكلة في حقل اختصاصك الذي تستطيع حلها جزئياً مع الرسومات بالحاسوب . عليك اختيار الخواص المميزة لمنظومة الرسومات الخاصة بك. أعطي وصفاً لمجموعة من الكيانات الأولية للرسومات (Graphical Primitives) التي تساعدك في حل المشكلة.

2.1 إن اختيار مجموعة كيانات أولية لمنظومات الرسومات تعتمد عادة على مبدأ التعامدية (Principle of Orthogonality) الذي نصه هو، ليس بمقدورنا بناء أي كيان أولي من الكيانات الأخرى. هل المنظومة التي تشتمل كياناتها الأولية على خطوط ورموز ومضلعات ودوائر تتفق مع هذه القاعدة؟

3.1 في برامج الحاسوب المكتوبة في لغات عالية المستوى، مثل فورتران أو لغة C، عادة تكون إحدى مدخلات المستفيد المطلوبة في صيغة سلسلة من الحرفيات. ماذا ستكون صيغ المدخلات لبرنامج يستخدم رسومات الحاسوب والتي ينبغي أن تؤخذ بنظر الاعتبار والتي تكون ملائمة؟

4.1 افترض لدينا شاشة نستطيع عرض 256 لون في آن واحد وعناصر الصورة لها أبعاد 512×512 . كم تكون حجم الذاكرة لاستخدام هذه العارضة بصورة كاملة. كيف يتغير حجم الذاكرة المطلوبة عندما تزاد دقة العارضة (عدد عناصر الصورة للصف الواحد أو العمود). إذا كان المطلوب إعادة رسم الصورة 60 مرة بالثانية، عند أي معدل يجب معالجة عناصر الصورة من قبل معالج العرض.

5.1 إحدى المجالات النشطة في أبحاث الحاسوب هي المعمارية المتوازية (Paralled Architectures) . في منظومة متوازية، توجد هنالك معالجات متعددة تعمل في وقت واحد على نفس برنامج المستفيد. أين تعتقد بالإمكان إدخال التوازي في رسومات الحاسوب.

الفصل الثاني

أفكار أساسية (Fundamental Ideas)



الفصل الثاني

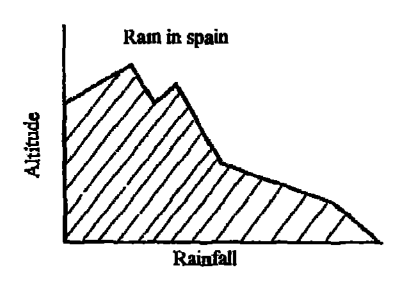
الأفكار الأساسية (Fundamental ideas)

مقدمة:

في هذا الفصل، سوف نتعرف على المفاهيم الأساسية لرسومات الحاسوب الحديثة، سيكون لبعض هذه المفاهيم أسس رياضية، وأخرى فلسفية وبعضها يمكنك التعرف عليها والأخرى ستكون جديدة. بصورة عامة، ستوفر هذه المفاهيم ليس فقط القاعدة في تطوير منظومة نواة الرسومات GKS، والذي سنقدمه في الفصل القادم، بل أيضاً أسس تطوير معظم منظومات الرسومات الحديثة.

1.2 إجراء راسم بياني بسيط (Simple Plotting Procedure)

سنبدأ الأخذ بنظر الاعتبار مخرجات إجراء الراسم البياني الذي سنتوسع في تفاصيله في الفصل الثالث. الشكل 1.2 يبين مخرجات نموذجية.



الشكل 1.2 مخرجات برنامج راسم بياني

يكتب البرنامج على شكل دالة راسم بياني يدعى 'Plot' والذي يتيح للمستفيد إدخال المعلومات التالية:

1- إحداثيات y,x لـ n من النقاط.

2- محاور معنونة (تمييز المحاور Label Of The Axes).

3- عنوان المخطط أو الراسم البياني "plot".

فيما يلي هيكل لهذا الإجراء الذي يحاكي (Simulate) ما قد تقوم به لو أردت رسم مخطط بياني باليد باستخدام ورقة وقلم:

```
Plot (n, x,y, xlabel, ylabel, title)
int n;   / *number of points */
float * x, * y ; / * x and y data array */
char * xlabel, *ylabel, * title ;
{
      draw- axes  (..................);
      plot – data (...............);
      label-axes (..............);
      draw-title  (...............);
      }
```

من أجل كتابة هذه الدالة، ينبغي أولاً أن تجلب الانتباه حول القضايا التالية:-

1- مع أي من الكيانات الأولية سنعمل ؟

2- كيف نقوم بوصف ومعالجة هذه الكيانات؟

3- كيف نقوم بوصف الصورة التي ترغب الحصول عليها؟

في هذا المثال البسيط، يمكننا التقييد باستخدام كيانين أوليين هما: قطع من الخطوط (Line Segments) وصفوف من الرموز (String of Characters).

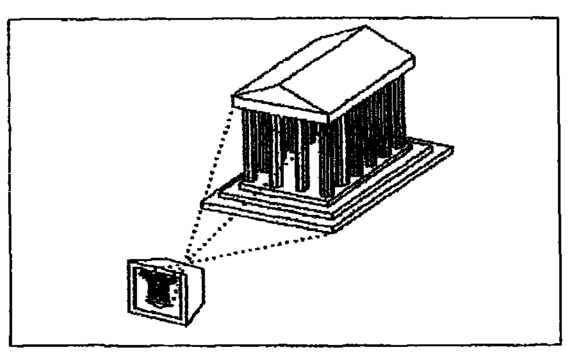
نستطيع استخدام قطع الخطوط لرسم المحاور والتوصيل بين نقاط البيانات (y,x). يكون من الضروري أن نقوم ببعض الحسابات لمطابقة للمخطط البياني ضمن المنطقة المستهدفة على العارضة حيث تحسب المواقع المختلفة لقطع الخطوط. لذا نحتاج إلى بعض الرياضيات للقيام بهذه الحسابات وسوف نتعرف إلى ما نحتاجه فيما بعد.

علينا أولاً حل مشكلة مفاهيمه قد تكون أكثر أساسية لتقرير كيفية رؤية عملية تركيب الصورة. في مثالنا لرسم المخطط البياني، لنبدأ مع البيانات. يتم إعطاء هذه البيانات كأزواج من الأرقام في منظومة إحداثيات تعتمد على المشكلة الأصلية التي

أنتجت هذه الأرقام. إذن لتكوين الصورة أو الرسم، ينبغي علينا تقييس البيانات (Scale the Data) لتطابق الجزء المستهدف من العارضة. أيضاً لربما نحتاج تقييس العناوين لتطابق الصورة . وبما أن الصورة في نهاية المطاف ستظهر على جهاز حقيقي كجهاز CRT أو راسم خطي، إذن ينبغي أن تكون هناك عملية تحويل إحداثيات من الوحدات المستخدمة في المشكلة الأساسية إلى وحدات جهاز الإخراج. ونظراً لأن الصورة المطلوبة تتألف من أجزاء متباينة، لذا تظهر الحاجة لإيجاد طريقة لوصف ومعالجة هذه الأشياء المنظورة (Objects) وذلك من أجل تركيب الصورة النهائية. إن هذا المتطلب هو متطلب مشترك في كل من الرسومات التقليدية والرسومات على الحاسوب.

2.2 تكوين الصورة *(Picture Formation)*.

كلاً من الرسومات التقليدية ورسومات الحاسوب يهتمان بتكوين الصور. سنقوم بالتوسع في تقنيات تكوين الصورة نحو دراستنا لرسومات الحاسوب. إن الصلة الموجودة بين الرسومات الحديثة والطرق الأكثر تقليدية في تكوين الصورة مثل التصوير الزيتي (Painting) والتصوير الضوئي (Photograph) قوية جداً . لنتأمل في عملية التصوير الضوئي كما مبين في الشكل 2.2.



الشكل 2.2 تكوين صورة

لدينا في هذه العملية شيئان متميزان قائمان بذاهما كيانين (Two Entities) أحدهما العالم حولنا، والذي يحتوي على أشياء منظورة كالبيوت والأشجار والأرض المعشوبة والآخر المشاهد لهذه الأشياء المنظورة وهي آلة التصوير (الكاميرا). ما هو موجود في

الصورة الضوئية سوف يتحدد فقط عندما نعرف كل شيء عن : أولاً آلة التصوير (مثلاً أين موقع آلة التصوير وأي نوع من العدسات عليها وفي أي اتجاه يشير)، ثانياً: الأشياء المنظورة (مثلاً مواقعها وواجهاتها وخواص سطوحها). إذا حركنا آلة التصوير، فالصور المسجلة على الفلم ينبغي أن تتغير، مثلما لو حركنا أي من الأشياء المنظورة.

إذن ناتج عملية تكوين الصورة هي ظهور الصورة على الفلم. تتضمن نفس العملية في حالة وصف تكوين الصورة من قبل رسام زيتي وما نجده على سطح صورة تلسكوب.

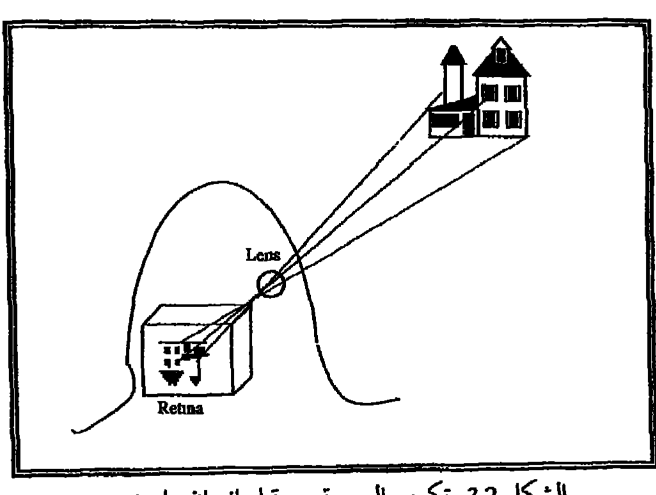
سنقوم بتوضيح مفاهيم الرياضيات (في الفصل الثامن ولا حاجة للفت النظر إليه الآن) المتعلقة بكيفية اشتراك الأشياء المنظورة والموجودة في عالمنا ثلاثي الأبعاد مع المشاهد الذي يأخذ موقع في هذا العالم لتكوين صورة ثنائية البعد.

تبدأ الرسومات التقليدية مع موجودتين (كينونتين) هما:

1- عالم الأشياء المنظورة (الأجسام).

2- المشاهد الذي يرغب في تكوين الصورة لهذه الأشياء المنظورة.

الشكل 3.2 يصور لنا عملية تكوين الصورة هذه إنسان راصد، بدلاً من آلة التصوير.

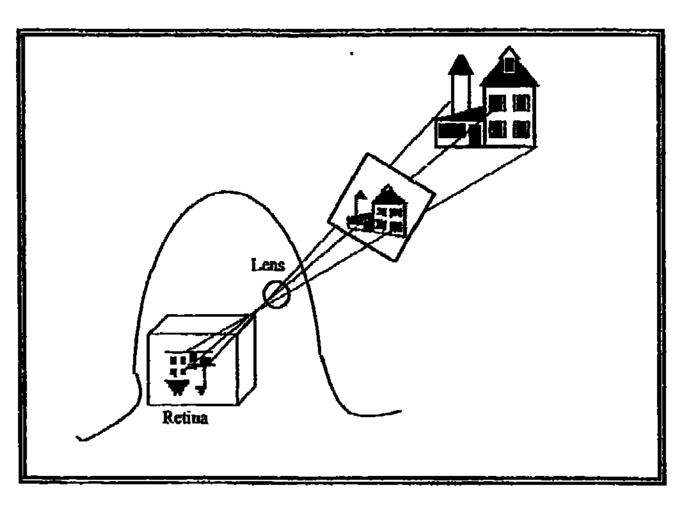


الشكل 3.2 تكوين الصورة من قبل إنسان راصد

لقد طرأ تغيير طفيف، هو أن بصريات العين حلت محل آلة التصوير. تتكون الصورة على شبكية العين، حيث يقع الكيان الحسي خلف العين. كل من الشبكية والفيلم خلف

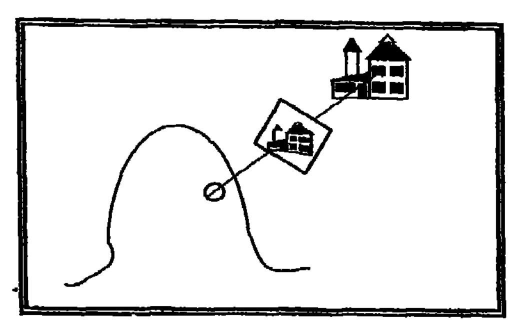
آلة التصوير لهما سعة محدودة. حيث لا يمكن للصورة المتكونة أن تمتد إلى ما لا نهاية. بالنسبة للرؤية البشرية ما نراه في الصورة هو كل شيء يقع ضمن مخروط الذي يكون رأسه (قمته) عند العين. وأما بالنسبة لآلة التصوير، مخروط الرؤية يحل محله هرم الرؤية، نظراً لكون مؤخرة آلة التصوير تأخذ شكلاً مستطيلاً.

الشكل 4.2 يبين نظرة مفاهيمه مختلفة قليلاً لعملية تكوين الصورة، حيث نرى الآن مستوى الصورة قد تم نقله إلى أمام آلة التصوير.



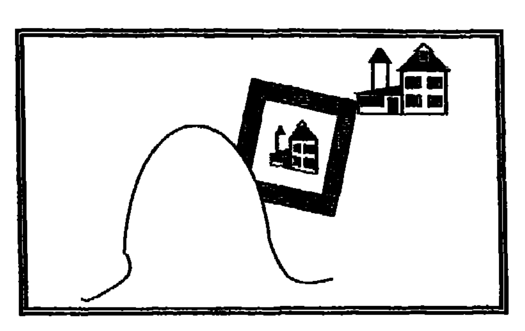
الشكل 4.2 تحريك مستوى الإسقاط

يكون الفرق الرئيسي هنا هو أن الصورة قد ظهرت غير مقلوبة وهذا غالباً يسهل على تفهم عملية تكوين الصورة. نستطيع أن نفكر بأن الصورة تتكون من رسم خطوط مستقيمة تبدأ من قمة هرم الرؤية (Viewing Pyramid) ، يدعى مركز الإسقاط (Center of Projection)، إلى جميع النقاط على الجسم أو الشيء المنظور. يدعى أي خط من هذه الخطوط، بخط الإسقاط (Projector)، الذي يتقاطع مع مستوى الصورة أو مستوى الإسقاط (Projection Plane)، الذي تقع عليه صورة نقاط الشيء المنظور، كما مبين في الشكل 5.2.



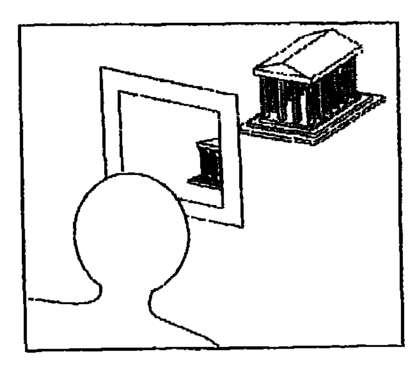
الشكل 5.2 صور وإسقاطات

أما في الشكل 6.2 يبين لنا صورة مرادفة إلى ما سبق.



الشكل 6.2 نوافـــــــذ

حيث نرى أن أسلوب الرؤية هنا يكون ببساطة مشابهاً إلى الرؤية من خلال نـــافذة. وما نراه من خلال النافذة ينبغي أن يكون محـــددا، وعندما نقوم بتحريك المشاهد أو النافذة، تظـــــهر صور مختلفة داخل النافذة (انظر الشكل 7.2).



الشكل 7.2 تغير النافذة

إن تركيب الصورة بالحاسوب يتبع نهجاً كثير التشابه إلى ما تم شرحه الآن والذي أصبح يعـــرف في مجال الرسومات بالحاسوب بمبدأ المناظرة بآلـــة التصوير الاصطناعيـــة أو التركيبيـــة (Synthetic Camera Analogy).

3.2 آلة التصوير الاصطناعية (التركيبية) (The Synthetic Camera)

1.3.2 فصل المشاهد عن الأشياء المنظورة (Separating The Viewer From The Objects).

عندما نقدم على أخذ صورة مع آلة التصوير أو أي طريقة أخرى لتكوين الصورة (مثلاً الرسم الزيتي، أو استخدام تلسكوب أو ببساطة النظر إلى شيء بالعين المجردة)، يمكننا اعتبار عملية التصوير اتحاد الأشياء المنظورة مع المشاهد. لم يكن واضحاً في الأيام المبكرة من عهد الرسومات في الحاسوب كيفية تطبيق هذا المفهوم في كتابة برامج الرسومات. كانت تتطلب لغات برمجة الرسومات ، أن يقوم المستفيد بوصف الصورة مباشرة من خلال دوال تحاكي عملية الرسم الحقيقية للصورة. لنرى قطعة برنامج لراسم قلمي (Pen Plotter) يبدوا بعض الشيء إلى ما يلي:

```
pen – up ( ) ;
move-pen (x1 , y1);
pen-down ( );
move – pen (x2, y2);
```

تكون وحدات الموقعين (x_1 , y_1) و (x_2, y_2) هي نفس وحدات الراسم القلمي (بالسنتمترات أو الإنجات). لقد أدى البرنامج بالراسم القلمي إنزال القلم. وبعدئذ، يتم تحريك القلم من (x_1,y_1) إلى (x_2,y_2) وهكذا يتم رسم قطعة الخط الواصل بين هاتين النقطتين، كما مبين في الشكل 8.2.

الشكل 8.2 نموذج راسم قلمي

لو عدنا إلى الشكل 2.2 نستطيع إدراك بعض المعضلات بسبب هذه الآلية التي تركت للمبرمج. تكون الأشياء المنظورة في عالم ثلاثي الأبعاد، كما هو الحال مع آلة التصوير. ومن أجل حساب المواقع (x_1, y_1) و (x_2, y_2) ، يتطلب من المبرمج إيجادها من خلال رياضيات الإسقاط أو تكوين الصورة. هكذا، من أجل رسم خط بسيط، عليه أن يعمل حسابات معقدة في المثلثات. إن ترك هذه الحسابات على المستفيد ليس فقط يتطلب منه إضافة كمية هائلة من عبارات البرمجة، بل أيضاً كانت طريقة تفتقر مفاهيمياً للتوجه نحو تطوير برمجيات الرسومات. لنأخذ على سبيل المثال، ماذا يحدث إذا حركنا آلة التصوير في مثل هذه المنظومة. ينبغي من برنامج التطبيق حساب إحداثيات جميع المواقع الجديدة وحسب وحدات الراسم القلمي. هكذا

نجد في مثال فكري بسيط، لو تركت آلة التصوير تجول حول المشهد، تصبح مهمة تحتاج إلى مجهود كبير.

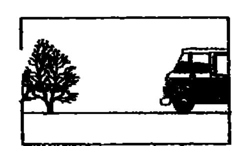
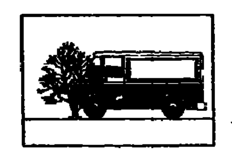
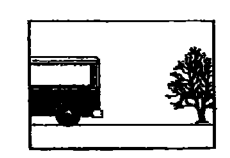
يكون البديل هو استخدام مبدأ المناظرة مع آلة التصوير الاصطناعية وتوفير منظومة رسومات تتعامل مع المشاهد والأشياء المنظورة بصورة منفصلة.

بالطبع، يجب إجراء الحسابات لإيجاد المواقع التي ستظهر فيها الخطوط على الراسم القلمي. مع ذلك، في المنظومات الحديثة تجري هذه الحسابات ضمن منظومة الرسومات (Graphics System). وهنا تكمن أهمية المناظرة بآلة التصوير الاصطناعية. إذ يقوم مبرمج التطبيقات بتحديد مواصفات مستقلة للشيء المنظور وشروط الرؤية. إن عملية تكوين الصورة ، تقوم بتوحيد هذه المواصفات، حيث تكون إحدى مهمات منظومة الرسومات.

إن الجزء الأساسي (البدن – Body) لبرنامج الرسومات يحتوي على نوعين من العبارات الرئيسية:

1. عبارات تصف المشاهد (Viewer).
2. عبارات تصف الأشياء المنظورة (Objects).

بالرغم من أننا لحد الآن لم نتعرف على دالة واحدة حقيقية للرسومات، لكن باستطاعتك تصور كيف يكون هيكل البرنامج الذي يقوم بإنتاج صور متشابه لشيء منظور متحرك كالسيارة ويُرى من قبل مشاهد موقعه على شيء منظور أيضاً متحرك كالدراجة. لهذا يمكن إظهار سلسلة من الصور كما هو مبين في متحرك كالدراجة. لهذا يمكن إظهار سلسلة من الصور كما هو مبين في الشكل 9.2.

الشكل 9.2 تحريك كل من الأشياء المنظورة والمشاهد

قد يبدوا البرنامج الآتي الذي تستخدم فيه عبارات مستعارة (Pseudocode) كالآتي:

```
describe – bicycle(  );      / *  the viewer */
describe – automobile (   ) ;   / * a moving object * /
describe – background (  );    / * non-moving objects */
```

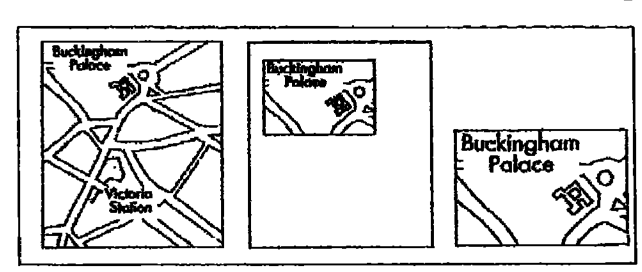
```
while (desire-new-images)
{
      move – bicycle (   ) ;
      move – automobile  (   ) ;
      output- new- image (   );
}
```

تقوم دالة إخراج صورة جديدة (output-new-image) إخبار منظومة الرسومات بأخذ المواصفات الحالية للشيء المنظور وتوحيد هذه البيانات لإنتاج صورة جديدة على العارضة. إن تفاصيل كيفية إجراء عملية التوحيد هذه تكون واضحة إلى برنامج التطبيقات.

2.3.2 مشاهدة ثنائية البعد: (Two – Dimensional Viewing)

تكون مشكلة المشاهدة سهلة في البعد الثنائي من حيث الأفكار والرياضيات بالإمكان اعتبار جميع الأشياء المنظورة موجودة في سطح مستوى وعالم ثنائي الأبعاد (Two Dimensional World) . إذن يتم تحديد قطع الخطوط بواسطة نقاط النهاية (x_1,y_1) و (x_2,y_2).

لا يوجد هناك تصور لشيء منظور (أو جزء منه) واقع في مستوى أبعد من شيء منظور آخر بالنسبة للمشاهد لأن جميع الأشياء المنظورة تقع في نفس المستوى. إذن تقتصر الرؤية على تحديد أي جزء من مستوى الشيء المنظور يستطيع المشاهد رؤيته. يمكننا تصور هذه العملية لو فرضنا وجود جدار بين المشاهد ومستوى الشيء المنظور، كما هو مبين في الشكل 10.2.

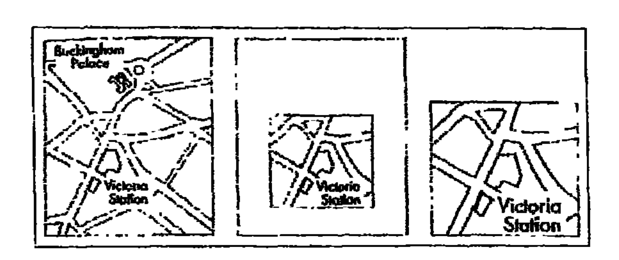


الشكل 10.2 مشهد ثنائي الأبعاد

أ- عالم ثنائي الأبعاد   ب- العالم من خلال النافذة   ج- الصورة

وفيه فتحة مستطيلة تدعى النافذة (Window) موضوعة في الجدار. ما قـــد يكـــون مرئي بالنسبة إلى المشاهد من خلال النافذة هو جزء من الصورة أو المشهد.

إذا تم توسيع فتحة النافذة، سيؤدي هذا إلى رؤية مساحة أكبر من مستوى الشـــيء المنظور، ولربما تظهر أشياء منظورة إضافية في الصورة. إذا نقلت الفتحة إلى مكان آخـر في الجدار لربما يظهر جزء مختلف من الشيء المنظور أو أشياء منظورة مختلفة من الصورة، كما هو مبين في الشكل 11.2.



الشكل 11.2 تغيير النافذة

أ- عالم ثنائي الأبعاد   ب- العالم من خلال النافذة   ج- الصورة

يكون هنالك إجرائين أساسيين في برنامج المستفيد هما:

1- دالة الخط (Function Line) التي تمكننا وصف أشياء منظورة بسيطة.

2- دالة النافذة (Function Window) والتي تمكننا تحديد نافذة مستطيلة وذلك بإعطـاء إحداثيات الزاويتين السفلى لليسار والعليا لليمين. هكذا يكون شكل البرنامج:

```
window (xmin, xmax, ymin, ymax);
line (x1, y1,  x2, y2);
line (x2, y2, x3, y3);
line (x4, y4, x5 , y5);
.
.
.

line (xN-1, yN-1 XN, yN);
```

3.3.2 استقلالية الجهاز(Device Independence).

المفهوم الآخر الذي نتج من مبدأ المناظرة بآلة التصوير الاصطناعية هـــو اســـتقلالية الجهاز. في مثالنا الأول، استخدمنا دالة واحدة هي move-pen. مع أن هذه الطريقة قـــد

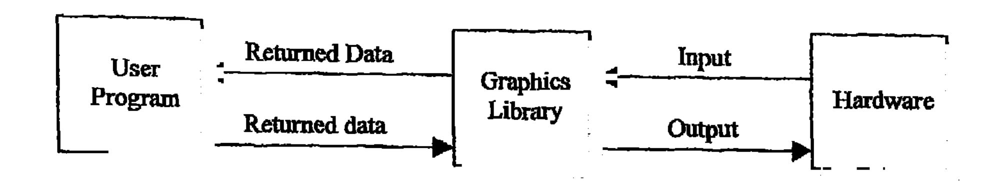
تبدوا أحسن من استخدام دالتين line, window، ولكن بتفحص عن قرب نرى هذا ليس صحيحاً. حيث تجمع الدالة move-open بين مواصفات الرؤية والأشياء المنظورة في أمـــر واحد منفرد. إن وحدات المعلمية (Parameters) y,x هـــي نفســـها وحـــدات جـــهاز الإخراج. لقد لاحظنا آنذاك أن المستفيد عليه إجراء كل الحسابات الضرورية لإيجاد هـــذه القيم، إذا استخدمت عارضة جديدة ذات قياسات مختلفة أو أخرى تعمل بوحدات مختلفة ينبغي تغيير كل هذه الحسابات في برنامج المستفيد.

لاحظ أوجه اختلاف هذه الطريقة مع ما يحدث عندما نســـتخدم window و line. هنا تكون المعلميات (Parameters) في هذه الدوال هي تلك التي تتعلق بالمسألة. أن عملية معالجة تكوين الصورة التي تأخذ هذه المعلميات (التي تكون في وحدات المسألة) وتقـــوم بتركيب الصورة (التي تكون في إحداثيات جهاز الإخراج)، كل هذا ينجز من قبل حزمـــة برمجيات الرسومات و ليس من شأن برنامج المستفيد.

هذا النموذج البرمجي (Programming Paradigm) ليس فقط يجنب المستفيد الشيء الكثير من الجهد، ولكن إضافة لكون برنامج المستفيد ليس فيه أي إشـــارة إلى وحـــدات جهاز الإخراج، لذا يدعى هذا البرنامج مستقل عن الجـــهاز (Device Independent). ليس فقط يمكن كتابة البرنامج بسهولة أكبر لأن وحداته تكون هي الوحـــدات الطبيعيـــة للمسألة الخاصة ولكن أيضاً يكون قابل للتنقل، لأنه لا يحتوي على شيء يخـــص جـــهاز الإخراج الذي قد يكون مستخدماً مع منظومة خاصة.

4.2 برمجيات مستقلة عن الجهاز *(Device-Independent Software)*

إن مبدأ المناظرة بآلة التصوير الاصطناعية جعل بالإمكـــان كتابـــة حزمـــة برامـــج للرسومات مستقلة عن الجهاز (Device Independent Graphics Package) في هـــذا البند، سنبحث بالتفصيل بنية هذه الحزمة من البرامج. وسوف نصف أنواع الدوال الـــتي يجب أن تتوفر ودلالات الاستقلالية لهذه الدوال عن الجهاز. يكـــون نموذجنـــا لبرنـــامج الرسومات هو برنامج تطبيقي مكتوب من قبل المستفيد مستخدماً فيه مكتبة الرســـومات (Graphics Library) ، كما هو مبين في الشكل 12.2.



الشكل 12.2 نموذج المبرمج

لقد تم التعرف على نوعين من الدوال هما: مواصفات الشـــيء المنظـــور وشـــروط الرؤية. في حزمة برامج كاملة، سوف نحتاج لمزيد من هذه الدوال. ستكون هنـــالك دوال تحكم (Control Functions) للتمهيد (Initialize) وإيقاف منظومة الرسومات وأجهزته. أيضاً ستكون هنالك دوال تسمح لنا تغيير الكيفية التي تعرض بهـــا الكيانـــات الأوليـــة (Primitives).

لحد الآن لم نبحث دوال الإدخال التي يجب أن تكون موجودة إذا أردنا كتابة برامـــج متفاعلة (برامج تفاعلية Interactive Programs) هذه الأنواع مـــن الـــدوال ســـتكون متوفرة، وحتى ضرورية، في معظم حزم برمجيات الرسومات. مع ذلك ، حالياً ســـنقتصر على دوال الرؤية والأشياء المنظورة، لأنها تكون الجزء الرئيسي من عبارات الرســـومات في برامج بسيطة.

إحدى فوائد مبدأ المناظرة مع آلة التصويـــر الاصطناعيـــة (Synthetic Camera Analogy)، هي تلك التي تسمح لنا التعبير عن إحداثيات هـــذه الـــدوال في وحـــدات المشكلة التي تناولها المستفيد. لأن هذه الدوال تقوم بوصف أشياء منظورة ومشاهد تتواجد في عالم المشكلة، لذا تعرف منظومة إحداثيات المستفيد هذا بمنظومة الإحداثيات الكونيـــة (World Coordinate – WC). في مثالنا السابق، تم التعبير عن معلميات (مقادير متغـــيرة القيم Parameters) في كلا الإجرائين line و window بالإحداثيات الكونية. توجد هناك ملاحظة مهمة، في معظم المنظومات، وهي يجب أن تكون الإحداثيات الكونيـــة أعـــداد حقيقية أو أرقام ذات الفاصلة السائبة (Floating Point)، لإتاحـــة الفرصـــة للمســـتفيد استخدام وحدات تتراوح من الميكرون (كما هو الحال في مجـــال تصميـــم VLSI) إلى الأمتار (كما هو الحال مع المخططات المعمارية). إن استخدام أعداد ذات الفاصلة الســـائبة قد تكون لها أثر مهم عند وضع المنظومة في حيز التطبيق.

عند نهاية الإخراج توجد الأجهزة الحقيقية (Physical Devices)، كمحطات طرفية للرسومات (Graphics Terminals) ورواسم خطية (Plotters). يجب إرسال إشارات صحيحة لهذه الأجهزة لتكون قادرة على إنتاج مخرجات للرسومات (Graphical Output). ولأن هذه الأجهزة تختلف الواحدة عن الأخرى، علينا أن لا نستغرب ذلك، بأن القيم التي تصف المواقع لهذه الأجهزة قد تختلف من جهاز إلى جهاز. على سبيل المثال، قد توصف مواقع على شاشة محطة طرفية للمسح الشبكي (Raster Terminal) بواسطة مواقع ذات أعداد صحيحة في مخزنه الانتقالي للصورة (Frame Buffer)، بينما الراسم القلمي قد يحتاج أن تكون مدخلاته مقاسة بالإنجات. أن منظومة إحداثيات أجهزة الإخراج والإدخال تدعى إحداثيات الجهاز (Device Coordinate-DC) وقد تكون مختلفة لكل جهاز.

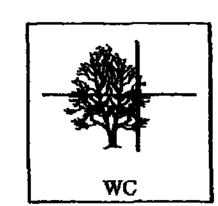
تحويل الإحداثيات يكون مطلوباً عند نقل القيم المحددة من قبل المستفيد (في WC) إلى إحداثيات الجهاز. تكون هذه إحدى المهمات الداخلية التي تقوم بها منظومة الرسومات. بصورة عامة ونظراً لكون المنظومة ستحتوي على أجهزة إدخال وإخراج متباينة، قد يبدوا أن حزمة برامج الرسومات تقوم داخلياً بمتابعة عديد من منظومات إحداثيات الأجهزة (DC Systems).

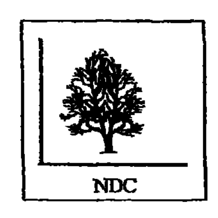
إن الأسلوب المتبع في معظم التطبيقات هو استخدام منظومة إحداثيات وسيطه تُعرف بمنظومة تعيير إحداثيات الجهاز أو إحداثيات قياسية (معيارية) الجهاز (Normalized Device Coordinate-NDC) لمعظم التطبيقات.

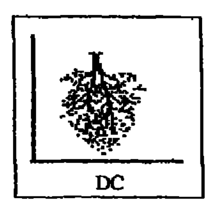
نستطيع أن نتفهم استخدام منظومة إحداثيات NDC لو أخذنا بعين الاعتبار فوائد تصميم حزمة برامج رسومات لجهاز إخراج واحد. يسمح لنا الجهاز الواحد بأن تكون لدينا عملية تحويل واحدة من إحداثيات المستفيد إلى إحداثيات الجهاز. يكون بالإمكان تطوير برمجيات لهذا الجهاز ونقله إلى حواسيب في مواقع أخرى . لذا الصور التي تم تمثيلها بدلالة منظومة إحداثيات الجهاز يمكن خزنها ونقلها بصيغة مصورة بدلاً من البرامج التي أنتجت الصور المنقولة. من الطبيعي، أن تكون المشكلة هي أن أجهزة الإخراج الحقيقية لها عدة منظومات إحداثيات مختلفة. مع ذلك، لو اخترنا جهاز واحد نظري أو افتراضي

(Theoretical or Virtual Device)، باستطاعتنا كتابة معظم البرمجيات الضرورية بدلالة هذا الجهاز الافتراضي. عادة يكون هذا الجهاز الافتراضي ذات مربع وحدة (Unit Square) في حيز منظومة تعيير إحداثيات الجهاز (NDC)، والذي تكون زاويته السفلى لليسار عند نقطة الأصل.

يتم تحويل القيم في وحدات الجهاز الافتراضي إلى قيم في وحدات منظومة إحداثيات الجهاز (DC) عند آخر مرحلة في المعالجة. ويتم تنفيذ هذا التحويل الأخير من منظومة إحداثيات NDC لجهازنا الافتراضي إلى منظومة إحداثيات DC بواسطة قطعة برمجية تعرف بوحدة إدارة الجهاز (Device Driver). في أي منظومة جيدة التصميم، كلما أضيف جهاز جديد، أو بدلت أجهزة قديمة، عندئذ يكون من السهل نسبياً إضافة وحدة إدارة جهاز جديد وبدون إجراء أي تغيير في الجزء الرئيسي من حزمة برامج الرسومات. الآن أصبح لدينا مستويين من التحويلات كما هو مبين في الشكل 13.2:







الشكل 13.2 التحويلات الإحداثية

الأول: التحويل من منظومة WC للمستفيد إلى منظومة NDC.

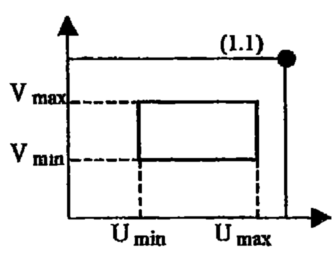
الثاني: التحويل من حيز NDC إلى DC. يوضح المثال جهاز حقيقي (Physical Device) يكون فيه ترتيب اتجاه y من الأعلى إلى الأسفل، هذا الاختيار ليس استثنائي في منظومات المسح الشبكي. عند الأخذ بنظر الاعتبار مدخلات الرسومات سوف تقلب عملية التحويلات ، حيث تكون المدخلات الفعلية في منظومةDC. لذا يتم تحويل هذه المدخلات أولاً إلى حيز NDC ومن ثم إلى WC.

5.2 النوافذ وبوابات الرؤية *(Windows and Viewports)*

سنقوم بنقل أو تحويل موقع تم تحديده في WC إلى نقطة مناظرة في NDC. تتوفر المعلومات الضرورية لبناء هذا التحويل في مواصفات النافذة التي تعرفنا عليها في البند 3.2.

قبل البدء باشتقاق المعادلات الضرورية، سنقوم بإضافة بعض المرونة إلى الرسومات من خلال استخدام ما يسمى بـ "بوابة الرؤية" (Viewport). سنأخذ بنظر الاعتبار عالم ثنائي الأبعاد حالياً.

غالباً ما نريد عرض أكثر من صورة واحدة على جهاز الإخراج. لقد رأينا في أمثلة سابقة، حيث نستخدم أجزاء من العارضة كواجهة بينية مع المستفيد، وأجزاء أخرى تستخدم للرسومات التي يقوم بتوليدها المستفيد. إن الحاجة في استخدام جزء من الشاشة فقط لعرض صورة معينة له أثر على عملية النقل (Mapping) من WC – إلى – NDC في عدة طرق. تكون إحدى هذه الطرق هو تغيير هذا النقل لكي يكون انتقال الأشياء المنظورة إلى جزء من فضاء NDC وبالتالي يتم نقله إلى الجزء المطلوب من عارضه جهاز الإخراج. قد تكون هذه الطريقة إلى حد ما غير عملية، لكونها تؤدي إلى التعارض مع مفهوم الاستقلالية عن الجهاز. توجد هنالك طريقة أفضل وذلك القيام بتحديد أي جزء من الشاشة نرغب تخصيصها لصورة معينة مستقلاً عن النافذة WC.



الشكل 14.2 بوابة الرؤية

تكون بوابة الرؤية عبارة عن مساحة جزئية (Subarea)من فضاء NDC في الشكل 14.2.

تم نقل نافذة في WC إلى بوابة نافذة في NDC.

وقد يتم تحديد بوابة الرؤية (Viewport)بواسطة الدالة التالية:

viewport (u-min, u-max, v-min, v-max);

وهو ثاني إجراء للرؤية. بخلاف دالة النافذة window تكون معلميات (Parameters) بوابة الرؤية محددة بفضاء NDC. القيم الأربعة يجب أن تحقق الشروط التالية:

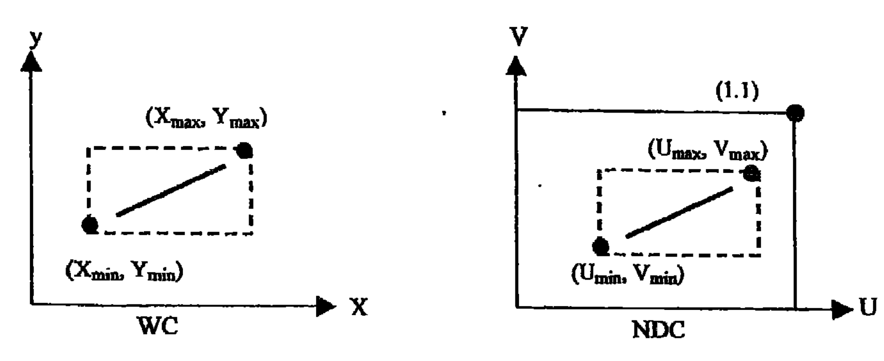
$$0 \le u_{min} < u_{max} \le 1$$
$$0 \le v_{min} < v_{max} \le 1$$

لأن الشاشة الافتراضية في فضاء NDC عبارة عن مربع وحدة (Unit Square) مع نقطة الأصل تقع في الزاوية السفلى لليسار.

الآن يمكننا اشتقاق الناقل (mapping) المتناظر مع قطعة البرنامج التالي:

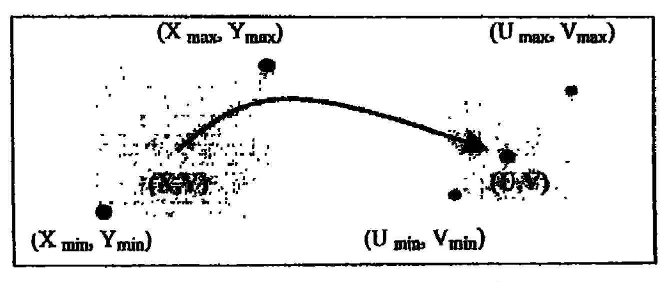
```
window (x-min, x-max, y-min, y-max);
viewport (u-min, u-max, v-min, v-max);
line (x1, y1, x2, y2);
```

مع أن الدالتين النافذة (window) وبوابة الرؤيا (viewport) ; يحددان عملية التحويل، لكن قطعة الخط الواصلة بين النقطتين (x_1,y_1) و(x_2,y_2) ينبغي أن تحول إلى قطعة خط واصلة بين نقطتين (u_1,v_1) و (u_2,v_2) في فضاء NDC كما مبين في الشكل 15.2.



الشكل 15.2 نقل قطعة خط

لزوج معطى من النافذة – البوابة (window-viewport)، سيكون التحويل نفسه على جميع النقاط في الإحداثيات الكونية (WC). نستطيع اشتقاق التحويل بافتراض نقطة عامة (x,y) في WC ومن ثم نقل هذه النقطة إلى صورتها (u,v) في فضاء NDC كما مبين في الشكل 16.2.



الشكل 16.2 النقل من WC إلى NDC

يجب المحافظة على نفس نسب الأبعاد في كلا الفضائين. هكذا ينبغي أن تكون عند نقل نقطة من وسط نافذة WC إلى نقطة تقابلها في وسط بوابة الرؤية. نستطيع كتابة معادلتي التناسب بصورة مستقلة لكل من الاتجاهين y,x:

$$\frac{x - x_{min}}{x_{max} - x_{min}} = \frac{u - u_{min}}{u_{max} - u_{min}}$$

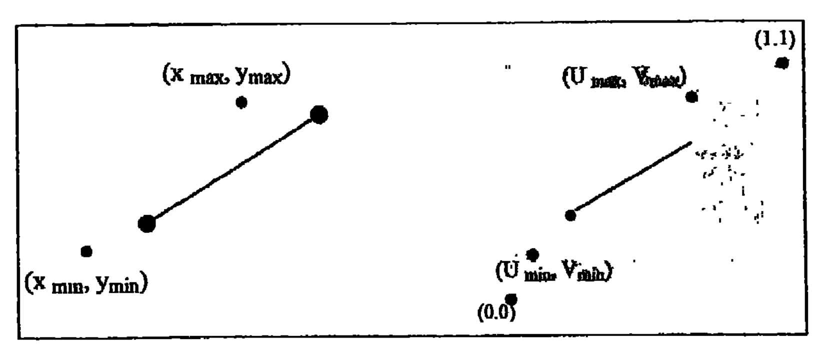
$$\frac{y - y_{min}}{y_{max} - y_{min}} = \frac{v - v_{min}}{v_{max} - v_{min}}$$

يمكن حل هذه المعادلات بدلالة (u,v) حيث نحصل على:

$$u = u_{min} + \frac{u_{max} - u_{min}}{x_{max} - x_{min}}(x - x_{min}),$$

$$v = v_{min} = \frac{v_{max} - v_{min}}{y_{max} - y_{min}}(y - y_{min}).$$

كل نقطة في أجراء الخط (line procedure) يتم تحويلها بواسطة هذه المعادلات لتحديد قطعة خط في فضاء NDC. لقد أخذنا أبسط حالة كما مبينة في الشكل 16.2، حيث تقع كلتا نقطتي نهاية قطعة الخط داخل النافذة. إذا كانت إحدى أو كلتا نقطتي النهاية لقطعة الخط تقطعان خارج النافذة، كما مبين في الشكل 17.2،

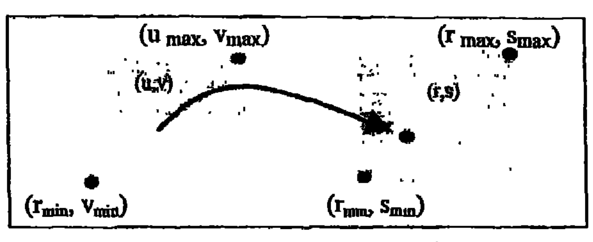


الشكل 17.2 التقليم (Clipping)

علينا أن نقرر ماذا سنفعل. في الواقع جميع منظومات الرسومات تهمل الجزء من الخط الذي يقع خارج النافذة عند عملية العرض. لهذا يمكننا القول أن الخط قد تم تقليمه

(Clipped) ضمن حدود النافذة. لاحقاً سنقوم بتطوير خوارزميات تتعلق بالتقليم (Clipping). في الوقت الحالي، عليك افتراض أن جميع الكيانات الأولية سوف تقلم ذاتياً، كإحدى المهمات التي يتم تأديتها من قبل منظومة الرسومات للمستفيد.

تستطيع استخدام زوج من المعادلات مشابه لما سبق لوصف عملية تحويل النقطة (u,v) إلى نقطة (r,s) في منظومة إحداثيات الجهاز (DC) لأي جهاز إخراج معين. على سبيل المثال، لنفترض لدينا جهاز عرض شبكي إحداثيات زاويته السفلى لليسار (r_{min}, s_{min}) وإحداثيات زاوية العليا لليمين (r_{max}, s_{max}) كما هو مبين في الشكل 18.2.



الشكل 18.2 النقل إلى إحداثيات الجهاز

إذن يكون النقل إلى إحداثيات الجهاز هو كما يلي:

$$r = r_{min} + \frac{r_{max} - r_{min}}{u_{max} - u_{min}}(u - u_{min}),$$

$$s = s_{min} + \frac{s_{max} - s_{min}}{v_{max} - v_{min}}(v - v_{min}).$$

توجد هناك نواحي عديدة تتعلق بهذه المعادلات قد تكون مهمة بالنسبة للمنفذ، مثلاً طريقة استخدام هذه المعادلات بكفاءة عالية وفي أي مرحلة من عملية إنتاج الصورة يتم تطبيقها. سوف نقوم بدراسة مثل هذه التساؤلات أو القضايا عندما نناقش موضوع التنفيذ في الفصل السادس.

6.2 تحديد الموقع *(Positioning)*:

الآن، أصبحت لدينا أساسيات لحزمة برامج رسومات، مثل دالة الخط لوصف الشيء المنظور ودوال النافذة وبوابة الرؤية لتحديد شروط الرؤية أو المشاهدة المطلوبة. قبل

الانتقال إلى وصف مظومة GKS كمنظومة رسومات فعالة يمكننا استخدامها لإنتاج برامج تفاعلية متطورة، سنرى من المفيد دراسة طرق مختلفة لإنجاز مهمة بسيطة مثل تحديد قطعة خط مستقيم. إن أحد أسباب التأمل أو التوقف عند هذه النقطة والأخذ بنظر الاعتبار بدائل أخرى هو، حالما نبدأ بمناقشة منظومة معينة (في حالتنا GKS)، قد يكون هناك ميل من القارئ ليفترض أن طريقة هذه المنظومة لإنجاز بعض العمليات هو الأسلوب الوحيد. مثل هذا الافتراض قد يحددنا بصرامة ويجعلنا لا نستمر بالبحث عن بدائل أخرى ولربما قد تكون أفضل لحل مشكلة معينة.

لنأخذ بنظر الاعتبار دالة الخط

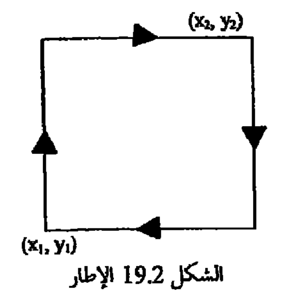
```
line (x1, y1, x2, y2);
```

لقد استخدمنا هذا الإجراء في تعريف قطعة الخط من (x_1, y_1) إلى (x_2, y_2).

توجد هنا ملاحظتين مهمتين هما:

أولاً: تم تحديد معلومات الموقع بقيم مطلقة في منظومة إحداثيات WC.

ثانياً: كل ما نحتاجه من المعلومات حول الخط هو استدعاء دالة الخط (line) مرة واحدة.



الشكل 19.2 الإطار

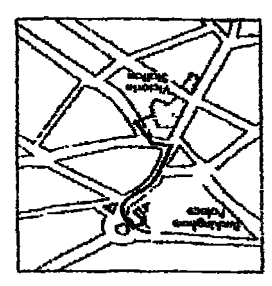
قد تبدوا هذه الاختيارات طبيعية بالنسبة لقطعة خط واحدة. ولكن ، لنأخذ بنظر الاعتبار مواصفات رسم مربع أو مستطيل كما مبين في الشكل 19.2.

تصف قطعة البرنامج التالية:

```
line (x1, y1, , x1, y2);
line (x1, y2, x2, y2);
line ( x2, y2, x2, y1);
line (x2, y1, x1, y1);
```

مربع، وذلك بتتبع حدوده باتجاه عقرب الساعة. نلاحظ هنا تكرار في ظهور بيانات النقاط، لربما تضايقنا قليلاً، وخاصة إذا أردنا وصف أشياء منظورة أكثر تعقيداً. أيضاً يمكننا أن نلاحظ هذا لو أردنا فعل شيء ما مثل إزاحة المربع إلى موقع جديد، سيتحتم علينا تغيير كل استدعاء للدالة line.

إن هذه الاعتبارات تقودنا إلى التساؤل فيما إذا يكون تحديد الموقع المطلق هو إمـا أن يكون طبيعياً على الأغلب أو طريقة مثلى لوصف المربع أو أي شيء منظور آخر.



الشكل 20.2 خارطة الشارع

الآن لنأخذ بنظر الاعتبار خريطة شارع كما في الشكل 20.2. يمكننا وصف المسار من أي نقطة إلى أية نقطة أخرى كسلسلة من الخطوط، معرفة كما في دالة الخطline . لكن وصف المسار بهذه الطريقة بالتأكيد ليس الطريقة التي نرغب فيها وصف المسار بالكلمات. على الأرجح، لو طلب منا أن نعطي تعليمات لشخص ما، قد تكون التعليمات قولاً وتشبه بعض الشيء إلى ما يلي :

اذهب باستقامة على مدى مجموعتين من الأبنية، ثم استدر يميناً، ومن ثم اذهب باستقامة على مدى ثلاثة مجاميع من الأبنية، ثم استدر يساراً...الخ. لاحظ هذه التعليمات كلها "نسبة إلى"، بدلاً من أن تكون مطلقة الصيغة. إن الأوامر النسبية ليست فقط تكون في معظم الأحيان طريقة أكثر طبيعية في وصف الشيء المنظور، بل أيضاً تكون ذا أهمية في سهولة التنفيذ في المكونات المادية والبرمجيات معاً.

إن تحديد الموقع النسبي يتم وصفه بمعلومات موقعية وبلغة "أين نحن الآن"، على سبيل المثال" تقدم إلى الأمام ستة خطوات" ، باستطاعتنا تصور الموقع الحالي (Current Position-Cp) كمنزلقة رسومات أو مؤشر الشاشة (Cursor) الرسومات في فضاء WC. في أي وقت، يعطي CP الموقع الحالي للمنزلقة. وبما أن CP يمثل موقع في فضاء WC، بالإمكان تمثيله في بعدين وكما يلي :

$$CP = (CP_x , CP_y)$$

وفي البعد الثلاثي علينا إضافة CP_z لكي نحصل على CP ثلاثي الأبعاد.

يمكن استبدال دالتنا للخط line بدالة: line- to (x, y)

يقوم هذا الإجراء بتعريف قطعة خط تبدأ من (CP_x, CP_y) وتنتهي عند (x,y). أيضاً يقوم هذا الإجراء بتغيير القيمة الحالية لـ(CP_x,CP_y) إلى (x,y)، لذا عند استدعاء الدالة line-to في المرة القادمة، سيبدأ من أين ما انتهت إليه آخر قطعة خط. الآن ظهرت لدينا

مشكلة لم تكن موجودة مع line. حيث لا يمكننا تعريف قطع من الخطوط المتتالية الغير متصلة. يكون الحل لهذه المشكلة بإضافة دالة ثانية وهي:

move – to (x,y)

ومهمة هذه الدالة هي تغيير (CP_x, CP_y) إلى (x,y). وبتعديل طفيف آخر سوف يزيد من مرونة استخدام هاتين الدالتين. لقد تم تعريف الدالتين move-to و line-to مع (x,y) معرفة بقيمها المطلقة. الاحتفاظ بروحية تحديد الموقع النسبي، يمكننا إذن إضافة دالة أخرى هي:

line-rel (dx, dy)

التي تعرف قطعة الخط مبتدأ من (CPx, CPy) إلى (CPx+dx, CPy+dy) مع تغيير الموقع الحالي CP إلى الموقع الجديد . هكذا يمكننا تعريف الدالة move-rel بنفس الأسلوب. الآن لنأخذ دالة بسيطة تدعى box (side) التي تعرف مربعاً مركزه عند CP مع ضلعه "side".

```
box (side)
float side;
{
      move-rel (-side /2, - side/2) ;
      line – rel ( side, o.o);
      line - rel (o.o, side);
      line – rel (-side, o.o);
      line – rel (o.o, - side);
      move- rel (side/2,side/2);
}
```

لاحظ ذلك الدالة تترك الموقع الحالي الأصلي (CP) بدون تغيير، قد تكون على الأرجح قاعدة عامة جيدة . لذا لا تحتاج هذه الدالة إلى أي تغيير في حالة تعريف مربعات تكون مراكزها نقاط مختلفة في عالم الإحداثيات. كل ما نحتاج عمله هو أن نسبق كل استدعاء للدالة box بـ move –to لكي نضع CP في الموقع المطلوب. سواء نستخدم box كدالة أو ما يعادلها من عبارات في برنامج، كل ما نحتاجه هو تغيير عبارة واحدة أما

لتحريك أو إعادة تعريف مربع. إن هذه السهولة في الاستخدام لم تكون ممكنة مع تحديد الموقع المطلق (Absolute Positioning) الذي سبق استخدامه في دالة الخط line.

إن منظومة CORE (Gspc79) ومنظومات أخرى عديدة تستخدم هذا النوع من تحديد الموقع النسبي. توجد هنالك بعض المشاكل مع مثل هذه المنظومة. لنرى ماذا يحدث في قطعة البرنامج التالي:

```
move- to (x,y);
window (x1,y1,x2,y2);
line-to (a,b);
```

هنا تم تثبيت الموقع الحالي CPبواسطة الدالة move-to، إلا أن شروط المشاهدة (Viewing) قد تغيرت قبل تعريف قطعة الخط تماماً بواسطة استدعاء الدالة line-to. ليس من الواضح كيف نقوم برسم قطعة خط تحتاج نقاط نهايتيه تحويلين مختلفين من النوع الذي سبق تطويره في البند السابق. إن هذا نوع واحد من التقييد المنطقي الذي يمكن أن تحدثه منظومة تحديد الموقع النسبي (Relative Positioning) . أما منظومة تحديد الموقع المطلق (Absolute Positioning)، كما هو معمول به في GKS، لا يمنع مبرمج التطبيق من استحداث منظومة خاصة به في تحديد الموقع وذلك باستخدام منظومة تحديد موقع معلوم.

هناك منظومة أخرى تقوم بتحديد الموقع النسبي ولها بعض الأهمية ومستخدمة في LOGO (Paper 81). إن منظومة الرسومات المستخدمة هنا تدعى رسومات السلحفاة (Turtle Graphics). تقوم هذه المنظومة باعتبار الموقع الحالي سلحفاة تتحرك حول العالم ومعها قلم مثبت في مؤخرتها السفلى. وباستطاعة السلحفاة أن تتحرك إلى الأمام وتستدير إلى اليمين أو اليسار. أيضاً نسمح للسلحفاة أن ترفع أو تخفض القلم. إن موقع السلحفاة يمثل الموقع الحالية (CP) وعند حركتها حول العالم تقوم بتوليد مخرجات الرسومات. باستطاعتنا أن نبدأ وذلك بإضافة دالة تمهيد (Initialization Function) تقوم بوضع السلحفاة في موقع محدد. الآن تكون لدينا منظومة رسومات تعتمد على ستة دوال:

```
initialize (x,y);
forward (distance);
```

```
right (degrees);
left (degrees);
pen-up(  );
pen-down (  );
```

في هذه المنظومة، نستطيع تعريف مربع (box) يبدأ عند موقع السلحفاة كما مبين في قطعة البرنامج التالية:

```
for (i = o; i < 4; i ++)
{
      forward (side) ;
      left (90.0);
}
```

إذا قمنا بتغيير قيمة طول الضلع side وعدد درجات زاوية الاستدارة وعدد مرات التكرار ، يستطيع نفس البرنامج توليد أشكال تقريبية للدوائر واللوالب.

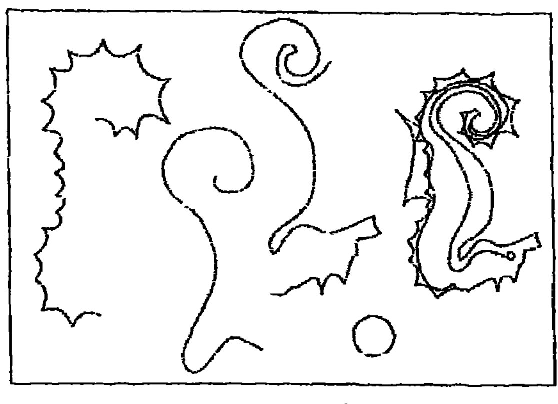
لربما بعض المنحنيات يمكن توليدها بسهولة أكبر باستخدام هذا النوع من منظومة تحديد الموقع بالمقارنة مع أما منظومة تحديد الموقع المطلق كما هو الحال في منظومة GKS أو منظومة تحديد الموقع النسبي كما هو الحال في منظومة CORE. كتمرين جدير بالاهتمام هو استحداث منظومة رسومات سلحفاتي من أحد هاتين المنظومتين (CORE, GKS).

7.2 نقاط، خطوط ومنحنيات (Points, Lines and Curves)

الآن لنتحول إلى مشكلة وصف الكيانات (Entities) التي سوف تستخدمها منظومة الرسومات. سنقوم بوصف هذه الكيانات في بعدين (Two Dimensions) في هذا البند، مع ذلك سوف نشير إلى ما يحدث عندما ننتقل إلى ثلاثة أبعاد (Three Dimensions) أينما يكون مناسباً.

إن عدد أبعاد (Dimensionality) الشيء المنظور يقصد به عدد القياسات الطولية المستقلة التي يمكن إجراءها عليه. على سبيل المثال، الخط له بعد واحد، لأن يمكننا قياس الطول بين أي نقطتين، لكن الخط نفسه ليس له عرض. (لاحظ ذلك، هنا نحن نتحدث

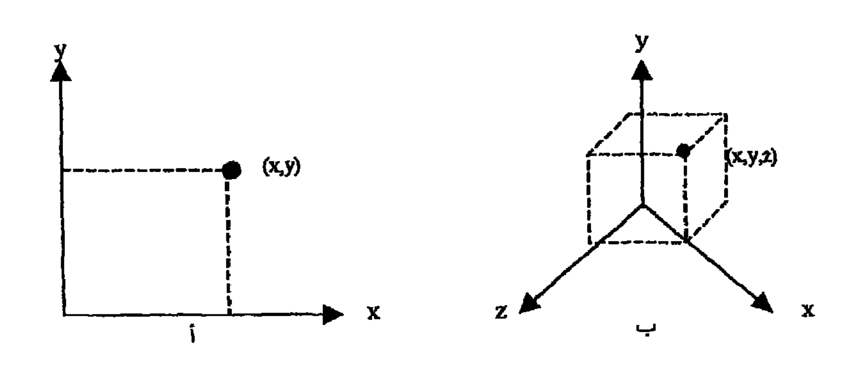
حول الخط في الرياضيات. الخط ممثلا على شاشة محطة طرفيه أو يرسم بالقلم له عـــرض غير صفري وإلا لا يمكن رؤيته). أيضاً المنحنيات المبينة في الشكل 21.2 لها بعد واحد.



الشكل 21.2 منحنيات

1.7.2 نقاط : (Points)

النقاط من الناحية الثانية تعتبر أشـــياء منظـــورة ذات البعـــث الصفـــري (Zero-Dimensional). النقطة لها موقع في الفضاء ولكن لا نستطيع إجراء قياسات طولية عليـــها. نظراً لاستخدامنا للنقاط في تعريف الخطوط والمنحنيات، لذا نحتاج أولاً طريقـــة لتمثيـــل النقطة كما مبين في الشكل 22.2.



الشكل 22.2 نقطة في: أ- بعدين ب-ثلاثة أبعاد

إن الطريقة الاعتيادية لتمثيل النقطة هي ، أما كزوج من البيانات (x,y) أو كمتجـه عمود (Column Vector) ذات عنصرين (عمود مصفوفة) $p=\begin{bmatrix} x \\ y \end{bmatrix}$

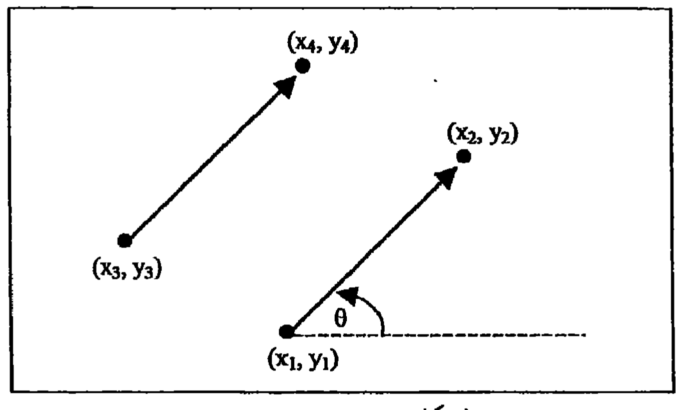
هذه الصيغة عادة تدعى متجه ثنائي الأبعاد (Two-Dimensional Vector) ، نظـراً لأننا قمنا بتمثيل النقطة في عالم ثنائي الأبعاد. إن هذا المصطلح لا يتناقض مع حقيقة كـون النقطة نفسها ليس لها أبعاد. أما النقاط في ثلاثة أبعاد فهي امتداد طبيعي للنقطة في البعدين حيث أن :

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

كما هو مبين في الجهة اليمنى من الشكل 22.2.

2.7.2 متجهات (Vectors).

في رسومات الحاسوب، غالباً ما تكون النقاط مقترنة مع المتجهات. أياً من التمثيلـين أعلاه للنقطة يمكن أن تدعى متجهة من قبل المتخصصين في الرياضيات. أما من وجهة نظر الفيزيائي فإنه يعرف المتجه "كيان له مقدار واتجاه معاً". لنفترض ، نحن نقوم برسم سلسلة من قطع بخطوط. كل قطعة لها اتجاه وطول أو مقدار. إذا تم تعريف المتجه v كقطعة خـط بين (x1,y1) و (x2,y2) كما مبين في الشكل 23.2.



الشكل 23.2 تعريف متجه

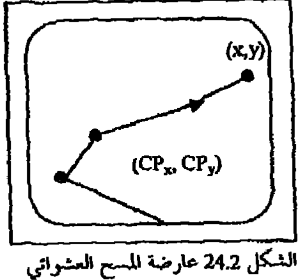
فإنه يمكن التعبير عن مقداره واتجاهه على التوالي كما يلي:

$$|V| = \sqrt{(x1 - x2)^2 + (y1 - y2)^2} \ ,$$

$$\tan\ \theta = \frac{y_2 - y_1}{x_2 - x_1}$$

هكذا نرى تم الحصول على متجه من نقطتين . لا زال هـذا التعريف لا يقنع الفيزيائي، نظراً لكون المقدار والاتجاه لا يثبتان موقع المتجه. قد تكون المتجهتان في الشكل 23.2 من وجهة نظر الفيزيائي متماثلين ولكن يبدو للعيان مختلفان جداً في الشكل .

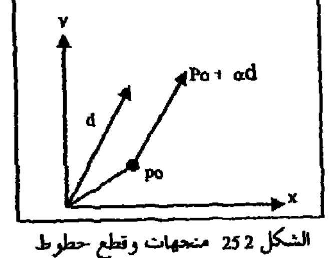
لكي نتجنب الإرباك ، تستخدم المتجهات في كثير من الأحيان لتعيين الاتجاهات بدلاً من المواقع. سنقوم بالتعقيب على هذا الاستخدام، خاصة عندما نتناول موضوع الرسومات ثلاثية الأبعاد (في الفصل الثامن).



الشكل 24.2 عارضة المسح العشوائي

يعود أصول استخدام المصطلح متجه في رسومات الحاسوب في وصف عارضات المسح العشوائي (Random-Scan Displays). لقد تم رسم سلسلة من قطع الخطوط كما مبين في الشكل 24.2.

عند كل نقطة نهاية لقطعة الخط ، ينبغي معرفة الاتجاه (أو المتجه) إلى نقطة النهاية اللاحقة من قبل المكونات المادية لرسم القطعة التالية. إن هذا المتطلب جعل الناس يصفون مثل هذه الأجهزة بـ"عارضات المتجه " (Vector Displays).



الشكل 25.2 متجهات وقطع خطوط

إن هذا الاستخدام للمتجهات والنقاط يكون مهماً بالنسبة لكل من المكونات المادية والرياضيات الأساسية معاً. لتأخذ بنظر الاعتبار الرسم البياني المبين في الشكل 25.2.

عندما نرغب أن نصف قطعة الخط من النقطة Po إلى نقطة أخرى P1، يتم معرفة اتجاه قطعة الخط هذا بواسطة المتجه d، حيث أن:

$$|d| = \sqrt{(x1 - xo)^2 + (y1 - yo)^2} \ ,$$

$$\tan\theta = \frac{y_1 - y_o}{x_1 - x_o}.$$

الآن يمكننا وصف النقاط الواقعة على قطعة الخط هذه بواسطة المعادلة:

$$p(\alpha) = p_0 + \alpha d$$

كلما ازدادت قيمة α ابتداءً من الصفر، تنبعث النقطة $p(\alpha)$ ابتداءً من Po وتسير في اتجاه d واصفتاً قطعة الخط هذه. يمكنك بسهولة حساب قيمةα الذي سيؤدي بالخط أن يمر من خلال النقطة P1 . بالإمكان استخدام هذه الصيغة ليس فقط في وصف قطعة الخط، بل أيضاً في توليد قطعة الخط بواسطة المكونات المادية (Hardware). إن هذا المثال هو التمثيل المعلمي للخط (Parametric Representation). إن أحد الاستنتاجات المستلة من هذا المثال هو، أن أفضل استخدام للمصطلح متجه يكون في وصف الاتجاه. الآن سنوجه اهتمامنا إلى دراسة موضوع الصيغ البديلة لتمثيل المنحنيات.

3.7.2 المنحنيات : (Curves)

في رسومات الحاسوب، غالباً ما نجد ليس من الضروري أن تكون الطريقة التي اعتدنا عليها في التفكير عن شيء هي أفضل طريقة. إن تمثيل المنحنيات هي بالضبط مثال لهذه الحالة. الخط هو حالة خاصة للمنحنى - الذي سوف نستخدمه دائماً. نظراً لكون أنواع التمثيل التي سوف نستخدمها للخطوط تكون مشابهة إلى تلك المنحنيات العامة، لذا سنقوم في وصف منحنيات عامة في بعدين ومن ثم تخصص إلى الخطوط.

4.7.2 منحنيات صريحة (Explicit Curves)

لربما يكون مألوفاً أكثر استخدام الصيغة الصريحة لتمثيل المنحني:

$$y = f(x)$$

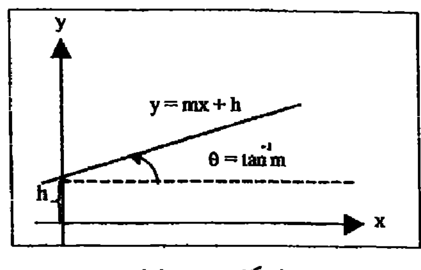
يكون المتغير المعتمد y معطى بصيغة صريحة بدلالة معادلة مستخدمة x كمتغير مستقل ومن الطبيعي، تكون صيغة التمثيل : $x = g(y)$ هي صيغة صريحة أيضاً. غالباً تكون المعادلات المألوفة للخط ودائرة مركزها عند نقطة الأصل كما يلي:

$$y = mx + h$$

$$y = \sqrt{r^2 - x^2}\,.$$

هاتان المعادلات توضحان بعض المشاكل مع التمثيل الصريح . في أحسن الأحوال، كلا المعادلتين ينتجان تمثيل غير كامل.

بالنسبة للخط المبين في الشكل 26.2 ، m هو ميل الخط (Slope) و h هـو الجـزء المقطوع من المحور y (y-Intercept). قد تنشأ مشكلة مع الخط العمودي.



الشكل 26.2 الخط

حيث يكون الميل مالانهاية، وبالتالي قـد لا توجد طريقة في استخدام المعادلة السابقة، مـن الطبيعي، نحن نستطيع قلب المعادلة ونستعمل:

$$x = \frac{1}{m}(y-h)$$

التي تعمل مع الخطوط العمودية ولكنـها تفشل مع الخطوط الأفقية. قد تحاول استخدام كلتا الصيغتين في وقت واحد، لكن هـذا سوف يكون غير مناسب، حيث ستكون البرامج ممتلئة بعبـارات "if else" لتحديـد أي الصيغتين يجب استخدامها. هذه المشكلة الخاصة تكون إحدى المشاكل المتنوعة الناتجة مـن الاعتمادية على المحور (Axis Dependence) والتي يتصف بها التمثيل الصريح للدالة. عندمـا يكون أحد المتغيرات معطى بدلالة المتغيرات الأخرى، لربما توجد بعـض قيـم للمتغـير المستقل بحيث لا يكون هناك حلاً.

من المعادلة الصريحة المعطاة للدائرة هنالك مشكلتين إضافيتين وهما:

أولاً: المعادلة تصف فقط نصف الدائرة الواقعة فوق المحور x وأما النصف الآخر كما يلي:

$$y = -\sqrt{r^2 - x^2}$$

الآن أصبحت لدينا معادلتين للدائرة، وهذا بالمعنى الدقيق، لا يكون تمثيلا صريحـاً. وهكذا يمكننا أن نستنتج من ذلك أنه لا يوجد للدائرة تمثيل صريح.

ثانياً: أما المشكلة الأخرى فهي أقل وضوحاً من الأولى. عندمـا نقـوم بتفحـص المعادلات، ندرك بأن كلا المتغيرين y,x يجب أن يكونا في المدى

$$-r \le x, y \le r$$

إذا لم نكن حذرين من بناء هذه المدايات داخل روتينات تعريف الدائرة في برامـج الحاسوب قد تواجهنا المشاكل، بالرغم من استخدامنا لهذه المدايات قد تسـتنفذ وقتـاً،

ولكن على الأقل هذه المدايات معروفة بالنسبة للدائرة. وقد لا تكون معروفة لمنحنيات بالغة التعقيد.

5.7.2 الصيغة الضمنية (Implicit Form)

من المألوف، أن نكتب معادلة الدائرة مركزها عند نقطة الأصل كما يلي :

$$x^2 + y^2 = r^2$$

هذا مثال للتمثيل الضمني للمنحنى (Implicit Curve)

$$g(x, y) = o$$

هذا التمثيل يتفادى كثير من المشاكل التي واجهتنا مع التمثيل الصريح. لهذه الطريقة فائدة أكثر من اختبار فيما إذا كانت نقطة معلونة تقع على المنحنى أم لا، بالمقارنة مع طريقة إيجاد نقاط واقعة على المنحنى.

في الحالات العامة، قد لا تكون لدينا طريقة تحليلية لإيجاد نقطة ما (x,y) تحقق معادلة ضمنية اختيارية (Arbitrary Implicit Equation). لذا يجب علينا اللجوء إلى تقنيات الطرق العددية (Numerical Analysis) والتي تكون عادة بطيئة جداً لتكون مفيدة في توليد منحنيات لعارضات الرسومات. مع ذلك، غالباً ما تستخدم الصيغ الضمنية مقترنة مع صيغ أخرى لتمثيل منحنيات عديدات الحدود (Polynomial Curves) والسطوح (Surfaces) وسنقوم بشرحها في الفصل العاشر. هنا، قد لا تكون عروض الزمن الحقيقي (Real-Time Display) قضية في موضوع دراستنا، وإن الحالة الخاصة لصيغ عديدة الحدود للمنحنى g (x,y) قد تبقي الحسابات ضمن حدود معقولة.

لنعود إلى معادلة الخط، ونقوم بدراسة تمثيله الضمني

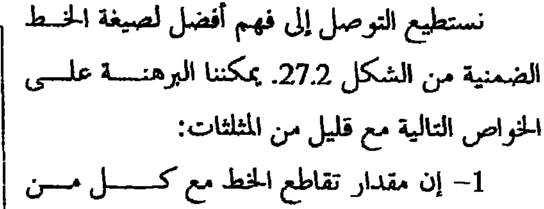
$$a x + by + c = o$$

وذلك لوصف عدد من خواصه المهمة. إذا كان أياً من a أو b صفراً، يكون لدينا خطاً موازياً أو عمودياً. سنفترض للسهولة أن كلا الحدين لا يكونا صفراً، حتى لو كانت الصيغة الضمنية قائمة لكلتا الحالتين. إذا قمنا بتقسيم طرفي المعادلة على a أو b نستطيع تحويل المعادلة في الصيغة الضمنية إلى الصيغة الصريحة. وإذا ضربنا المعادلة بأي مقدار ثابت غير صفري نحصل على:

$$sax + sby + sc = a`x + b`y + c` = o,$$

والتي تكون مرادفة للصيغة الضمنية. وهكذا، بالرغم من أن الصيغة الضمنية لها ثلاث معلميات (Parameters)، فعملية الضرب بـ s تبين لنا أنه بالإمكان جعل معلميه واحدة اختيارية بدون تغيير الخط. غالباً نحن نختار s بحيث إن الصيغة المعبرة تملك

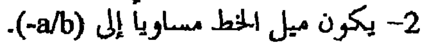
$$(a`)^2 + (b`)^2 = 1$$



نستطيع التوصل إلى فهم أفضل لصيغة الخط الضمنية من الشكل 27.2. يمكننا البرهنة على الخواص التالية مع قليل من المثلثات:

1- إن مقدار تقاطع الخط مع كل من المحورن y,x يكون عند المسافة (- c/a) و (-c/b) من نقطة الأصل على التوالي.

الشكل 27.2 خط معروف ضمنياً



2- يكون ميل الخط مساوياً إلى (-a/b).

3- جميع الخطوط مع نفس الميل تكون متوازية.

4- أي خط ميله مساوياً إلى b/a يكون عمودياً على الخط الأصلي .

لنأخذ بنظر الاعتبار مسألة إيجاد أقصر مسافة من نقطة الأصل إلى الخط:

$$a x + by + c = o$$

نحن نعلم بأن أقصر مسافة من نقطة إلى خط هو العمود بين تلك النقطة إلى الخط.

وبما أن ميل الخط في مثالنا هو a/b - ، إذن ميل العمود على الخط سيكون b/a. ونظرا لكون هذا العمود مرسوم من نقطة الأصل على الخط، لذا سوف يحقق العمود المعادلة التالية:

$$b x - a y = o$$

الآن أصبحت لدينا معادلتين في مجهولين، يمكن حلهما لنحصل على نقطة التقاطع:

$$x = \frac{-ac}{a^2 + b^2} \quad ، \quad y = \frac{-bc}{a^2 + b^2}$$

إذن تكون المسافة من نقطة الأصل إلى الخط هي:

$$d = \sqrt{x^2 + y^2} = \frac{c}{\sqrt{a^2 + b^2}}$$

إذا كان الخط قد تم تعيره $(a^2+b^2 = 1)$ نحصل على بعض النتائج المهمة وتصبح نقطة التقاطع:

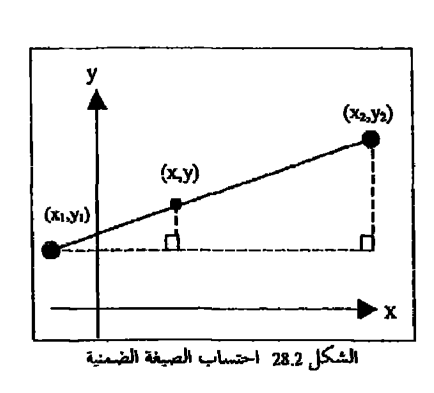
$$x = -ac$$
$$y = -bc$$

وتبعد مسافة c من نقطة الأصل. إذا تم تعير الخط في المعادلة الأصلية وكانت صيغة المعادلة

$$ax + by + c = o$$

فإن النقطة (a, b) تعرف عمود وحده (Unit Normal) من نقطة الأصل وتمثل c المسافة إلى الخط. بالنسبة للخطوط غير المعيارية، يكون الفرق الوحيد هو ضرب المسافة cبالثابت $(1/\sqrt{a^2 + b^2})$ لنحصل على المسافة من نقطة الأصل إلى الخط.

إن العلاقة في الصيغة الضمنية بين خط وعموده تبرز كثيراً في رسومات الحاسوب. وفي حالة ثلاثة أبعاد، تكون هناك علاقة مشابهة بين مستوى وعموده.



الشكل 28.2 احتساب الصيغة الضمنية

إن الصيغة الضمنية للخط تنشأ بطريقة طبيعية في رسومات الحاسوب. في معظم تطبيقات الرسومات، تكون اهتماماتنا ليس بالخط ولكن بالأحرى بقطعة الخط – هو جزء من الخط الواصل بين نقطتين، بالمقارنة مع خط طوله ما لانهاية. أما بالنسبة للصيغة الصريحة (Explicit Form ، نستطيع استخدام الرسم المبين في الشكل 28.2.

لحساب الميل والتقاطع من نقطتي النهاية (x1,y1) و (x2,y2) . لاحظ من تشابه المثلثات، يكون الميل

$$m = \tan\theta = \frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}.$$

نستطيع ضرب طرفي المعادلتين على اليمين للحصول على التمثيل الضمني التالي

$$(x_2 - x_1)\, y - (y_2 - y_1)\, x + x_1 y_2 - x_2 y_1 = 0 .$$

بصورة خاصة ، تكون هذه الصيغة مهمة بالنسبة لرسومات الحاسوب لأنه إذا أردنا تغيير إحدى نقطتي نهاية قطعة الخط، كما قد نرغب القيام به في تطبيقات متفاعلة (Interactive Application) ، فإنه تكون لدينا طريقة مباشرة لتغيير جميع قطع الخطوط التي تستخدم نقطة النهاية التي تم تغييرها.

هذه ليست نهاية القصة، إن الصيغة الضمنية غير مفيدة وخاصة للمنحنيات العامة، وحتى للخطوط، لأنها تحتوي على بعض المشاكل وهي:

أولاً: بالرغم من أن الصيغة الضمنية تدعم كلا من الخطوط العمودية والموازية، إلا أن بدون الاختبار لا نستطيع التأكد من أن لدينا إحدى هذه الحالات.

ثانياً: مشكلة أخرى، غالباً ما نرغب معرفة فيما إذا كانت نقطة معلومة تقع على قطعة الخط الواصل بين اثنين من النقاط الأخرى. قد تنشأ مثل هذه المشكلة عندما نحاول تحديد فيما إذا قطعة خط تقع داخل النافذة. الصيغة الضمنية المشتقة سابقاً قد لا تكون مفيدة وخاصة للوصول إلى هذا التحديد.

ثالثاً: قد تنشأ مشكلة أخرى عندما تريد إيجاد أفضل صيغة لتمثيل منحني من حيث توليد النقاط الواقعة عليه بواسطة المكونات المادية أو البرمجيات.

6.7.2 الصيغة المعلمية (Parametric Form)

إن التمثيل المعلمي أو الوسيط (Parametric) يتجنب كثير من هذه المشاكل. بدلاً من أن تكون هنالك معادلة واحدة تصف المنحني، نستخدم معادلتين من الصيغة الصريحة لها معلمية تدعى 't':

$$x = x(t)$$
$$y = y(t)$$

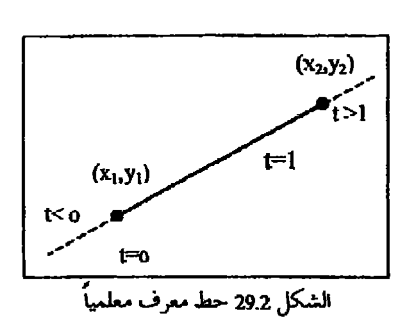
على سبيل المثال ، بالنسبة للخط الواصل بين نقطتين (x1,y1) و (x2, y2)، يمكننا استخدام المعادلتين التاليتين:

$$x = (1-t)\, x1 + tx2$$
$$y = (1-t)\, y1 + ty2$$

يمكننا التحقق من صحة هذه المعادلات وذلك بحذف t وتحويلها إلى صيغة صريحة.

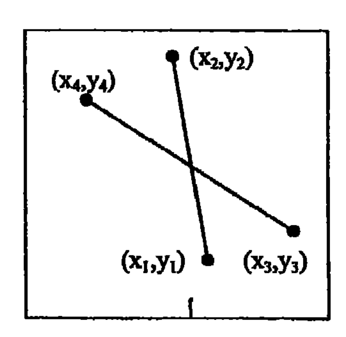
إن هذا الزوج من المعادلات ليس التمثيل المعلمي الوحيد للخط، ولكن يعتبر من الصيغ المفيدة، وذلك لسهولة حساب حدود الطرف الأيمن للمعادلة. لاحظ على سبيل المثال، عندما تكون $t=0$ تعطينا المعادلات إحداثيات النقطة (x_1,y_1)، وعندما تكون $t=1$ نحصل على النقطة (x_2,y_2)، وباستطاعتنا استخدام هذه الصيغة المعلمية لإجراء عمليات متنوعة تشمل قطع الخطوط. إذا اعتبرنا t كمتغير زمني، نستطيع توليد سلسلة من النقاط على قطعة الخط وذلك بزيادة t بخطوات صغيرة . يمكن أن تكون هذه المعادلات أساس في استخدام مكونات مادية لتوليد الخط (Hardware Line Generator).

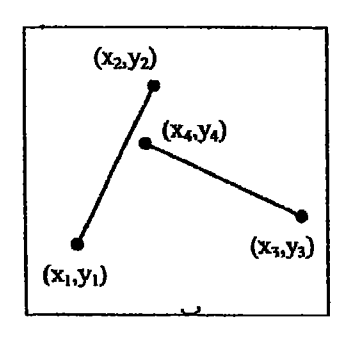
عند تغيير قيمة t من صفر إلى واحد، يتم توليد جميع النقاط على قطعة الخط الواصلة بين النقطتين (x_1,y_1) و (x_2,y_2). بالنسبة لقيم t التي تقع خارج مدى $(0,1)$، لازالت النقاط المولدة تقع على الخط، لكنها ليست بالضبط على القطعة المحددة من الخط، كما هو مبين في الشكل 29.2.



الشكل 29.2 خط معرف معلمياً

هذه الملاحظة تقودنا إلى إمكانية استخدام الصيغة المعلمية في اتخاذ قرار فيما إذا قطعتين من الخطوط يتقاطعان كما بين في الشكل 30.2.





الشكل 30.2 تقاطع قطع الخطوط

أ- قطع متقاطعة ب- قطع غير متقاطعة

قد تنشأ مثل هذه المشكلة، كما لاحظنا، عندما نحاول اتخذا قرار فيما إذا قطعه خط تقع داخل حدود النافذة وأيضاً في عدد من الحالات أخرى. إن أخذ مثل هذا القرار

أصعب بكثير من تحديد فيما إذا خطان يتقاطعان، نظراً لأن الخطوط تتقاطع طالما لا تكون متوازية. نستطيع الحصول على طريقة مباشرة وذلك بكتابة قطعتي الخط في الصيغة المعلمية. يتم تحديد قطعة الخط الأولى بواسطة :

$$x = (1-t)\,x_1 + t x_2$$
$$y = (1-t)\,y_1 + t y_2$$
$$0 \le t \le 1.$$

وأما قطعة الخط الثانية، سوف نستخدم المعلمية s للحصول على صيغة مشابهة كما يلي:

$$x = (1-s)\,x_3 + s x_4$$
$$y = (1-s)\,y_3 + s y_4$$
$$0 \le s \le 1.$$

يتم تحديد نقطة التقاطع بإيجاد زوج من (s,t) الذي يجعل كلا المجموعتين من المعادلات تحصل على نفس إحداثيات النقطة (x,y). وبواسطة وضع y,x المتناظرة في المعادلات تساوي إحداهما الأخرى، تكون قد حصلنا على معادلتين في المتغيرين t,s كما يلي:

$$(x_4 - x_3)\,s + (x_1 - x_2)\,t = x_1 - x_3$$
$$(y_4 - y_3)\,s + (y_1 - y_2)\,t = y_1 - y_3$$

من السهولة برهان ذلك، طالما هذه الخطوط غير متوازية، نستطيع حل هذه المعادلات لإيجاد t,s. إذا كان t,s كليهما بين الصفر والواحد، فالقطعتان تتقاطعان، وإلا تتقاطع الخطوط خارج الفترة.

8.2 بعض الاعتبارات في التنقلية *(Portability Consideration)*

في هذا البند الأخير سنتعرف على بعض المبادئ الأولية المتضمنة في هندسة البرمجيات (Software Engineering) لمنظومة رسومات مستقلة عن الجهاز (Device-Independent Graphics System) . تكون هذه المبادئ مشتركة في معظم المشاريع الكبيرة وبالتالي قد تكون مألوفة لديك، مع ذلك نرغب التأكيد على نقطتين:

أولاً: الفصل بين المستفيد من حزمة البرامج المستقلة عن الجهاز والمنفذ. مع أن مفهوم كتابة برمجيات مستقلة عن الجهاز باستخدام حزمة برامج رسومات قياسية (Standard Graphics Package)، فكرة جيدة وكبداية لنا في كتابة برامج، ولكن

سنواجه صعوبات وهي، من أجل تحقيق استقلالية عن الجهاز غالباً ما تكون على وجه التحديد غير ممكنة. كلما نقترب من خصوصيات تنفيذ حزمة البرامج المستقلة عن الجهاز، عادة يزداد قلقنا حول الاعتمادية على الجهاز.

ثانياً: أما القضية المهمة الأخرى هي تلك التي تتعلق بتتبع الخطأ وتدقيقه. سوف نقوم بمناقشة ضرورة تتبع الخطأ وسندرس مضمونه في كتابة برامج المستفيد.

1.8.2 الوظيفية ضد الشكلية (Functionality Versus Format)

عادة تكون المواصفات القياسية مواصفات وظيفية ، حيث تقوم بتحديد ماذا سيعمل القياسي (Standard)، مثلاً في حالة قياسي منظومة GKS، ماذا يكون محتوى القياسي من الإجراءات والدوال. هناك معلومات أخرى، مثلاً أية أخطاء سيتم تشخيصها وما هي أصغر مجموعة من أنواع الخطوط متوفرة، كل هذا هو جزء من القياس أيضاً.

لا تزال توجد هنالك الكثير من الأمور تحتاج إلى التعريف قبل أن يبادر المنفذ في بناء حزمة البرامج. على سبيل المثال، لنأخذ بنظر الاعتبار الإجراء أو الدالة الأساسية لتعريف قطعة الخط (Basic Line – Segment Defining Procedurs) تدعى متعدد الخطوط (Polyline) الموجود حالياً في معظم المنظومات. تقوم هذه الدالة بتعريف سلسلة من قطع الخطوط ابتداءً من النقطة الأولى إلى النقطة الثانية ومن النقطة الثانية إلى النقطة الثالثة وهكذا. قد تخبرنا المواصفات الوظيفية كأن هذه الدالة لها اثنين من المدخلات: عدد النقاط يكون عدد صحيح أكبر من واحد والنقاط إحداثياتها كونية أي في WC. مع أن هذا قد لا يكون واضحاً، يكون هنالك نوعان من الأخطاء المحتملة فقط:

1- قد لا تكون منظومة لرسومات في حالتها المناسبة عند استدعاء دالة متعددة الخطوط(polyline) . مثال ذلك، لربما لم يتم تمهيد (Initialized) المنظومة بصورة صحيحة.

2- عدد النقاط قد تكون غير كافية – يعني أقل من اثنين.

هذه المعلومات لا توفر لنا اسم الدالة عند الاستدعاء (function call) ، الذي يعتبر بالتأكيد جزء مهم من المعلومات عند كتابة برنامج. توجد هنالك عدة أسماء محتملة وواضحة للترابط (Binding) مع لغة C. نستطيع تبني الاسم المختصر بلغة فورتران GPL،

ولكن هذا لا ينسجم مع مفهومنا الحاضر للأسلوب الجيد في البرمجة مع لغة C. باستطاعتنا استخدام أسماء واضحة مثلاً أما polyline أو poly-line في الحقيقة، تم الاختبار من قبل لجنة GKS القياسية باستخدام الاسمgpolyline حيث استخدم فيه حرف البادئة g كما هو الحال مع جميع دوال GKS، أما في حالة منظومة PHIGS فالاسم المستخدم هو ppolyline. سوف نفترض أن المنظومة هي GKS مع أن نفس القرارات تنطبق على أي منظومة.

نستطيع الآن إلقاء نظرة على المعلميات (Parameters). أولا لنأخذ العدد الصحيح، هل أن العدد هو من النوع الطويل (long type) أو من النوع القصير (short type)؟ لو جعلنا العدد من نوع "int"، هنا تبقى مسؤولية التطبيقات المختلفة مستخدمة فيها أطوال مختلفة لـ int. يكون الحل الاعتيادي لهذه المشكلة هو تعريف أنواع بيانات لتطبيقنا، عادة تظهر هذه في ملف include. سوف تستخدم برامجنا نوع البيانات Gint فعلى سبيل المثال:

Gint num – pt;

للدالة gpolyline. يستطيع المنفذ أن يقرر كيفية تنفيذ نوع البيانات Gint والأنواع الأساسية الأخرى Gfloat و Gchar. بالإمكان وضع هذه المعلومات في gks.hأو في ملف include آخر، أيهما كان ، فسيظهر شيء مشابه إلى ما يلي:

```
typedef int Gint;
typedef float Gfloat;
typedef char Gchar;
```

في نموذج تطبيقي. يتم بناء جميع أنواع البينات الأخرى في GKS على أساس هذه الأنواع الثلاثة.

نعود إلى وظائف دالتنا polyline، حيث نجد موضوع تعريف مصفوفات البيانات لهذه الدالة أكثر تعقيداً. في لغة مثل فورتران (Fortran) توجد هناك عدة إمكانات، بينما في لغة C توجد إمكانات أكثر. نظراً لكون منظومة GKS هو قياسي ثنائي الأبعاد، فالنقطة في البعد الثنائي يجب أن تكون من نوع البيانات الأساسية، وحيث أننا نعمل مع الإحداثيات الكونية WC، لذا يجب أن نستخدم النوع الأساسي Gfloat. يتم تعريف النقطة في الترابط مع لغة C بواسطة typedef كما يلي:

```
typedef struct
{
    Gfloat x;
    Gfloat y;
}Gpt;
```

حيث تكون مدخلات الدالة polyline عبارة عن مؤشر إلى مصفوفة نقاط، هذا يعني

```
Gint num-pt;
Gpt array-pt [NUM];
:
:
gpolyline (num – pt, array-pt);
```

هكذا يتألف الترابط بلغة C من أسماء الدوال مع مدخلاتها ومخرجاتها وأنواع البيانات الضرورية مع إدراك ذلك، أن الأنواع ذات المستوى الأدنى (Lowest-Level Types) مثلاً، Gint، بالإمكان تحديدها محلياً (Locally). يشار إلى هذه المعلومات في بعض الأحيان بالشكل القياسي (Format of Standard).

2.8.2 بدائل افتراضية واختيارات (Defaults and Choices)

إن عدد من المعلميات التي قد تكون مدخلات لدوال رسومات يفضل تعريفها إما بواسطة الأنواع القابلة للعد (enumerated types) أو بواسطة استخدام وسيلة التعريف "define" في لغة C. على سبيل المثال ، قد يحتاج نموذجنا على الأقل إلى أربعة أنواع مختلفة من الخطوط هي: الخط المتصل (Solid)، والخط المتقطع (Dashed) والخط المتقطع – المنقط (Dashed-Dotted). يتم تحديد هذه الأنواع بأعداد صحيحة1، 2، 3، 4. بدلاً من استخدام هذه الأرقام السحرية، عادة نفضل استخدام شيء مماثل إلى ما يلي:

```
#define  GLN – SOLID      1
#define  GLN – DASH       2
#define  GLN – DOT        3
#define  GLN – DOTDASH  4
```

تكون مثل هذه الطريقة مفيدة خاصة عندما تكون لدينا أنواع أكثر من الحد الأدنى الذي يحتاجه نموذجنا.

إن طريقة العد للأنواع تكون مفيدة للمعلومات التي تكون متقطعة (Discrete). على سبيل المثال ، تقوم منظومة GKS بتعريف أربعة أنواع لملأ مساحات مجسمة (Solid Areas) من خلال العد للأنواع وكما يلي :

```
typedef enum
{
        GHOLLOW, / * draw only the boundary */
        GSOLID, / * fill with a solid color */
        GPAT, / * fill with a pattern of colors */
        GHATCH, / * fill with cross hatched lines */
} Gfill – int- style;
```

سنقوم باستخدام صفوف من الرموز الكبيرة (Uppercase Character Strings) فقط للأنواع القابلة للعد وصفوف الرموز المصرح بها في عبارة التعريف "de fine" بصورة عامة، هذا الاختيار هو أمر يتعلق بأسلوب البرمجة الجيدة وليس أمر خاص يتعلق بمنظومة الرسومات.

إن استخدام القيم الافتراضية (Defaults) بالنسبة لكثير من المعلميات كالألوان الأمامية والخلفية للصورة ونوع الخط، قد يتم تحديدها في القياسي أو قد يترك تحديدها للمنفذ. إذن، لو أردنا أن نحصل على برامج مستقلة عن الجهاز، ينبغي تحديد قيمنا الافتراضية.

لو جمعنا هذه القواعد والتوجيهات معاً، لربما نتوقع أن نرى عبارات برمجة مشابهة بعض الشيء إلى ما يلي عند بداية برنامج في منظومة GKS:

```
#include < gks.h > /* typedefs defined by standard */
#include < local – gks- typedefs.h>
#include < local – gks-defaults.h>
```

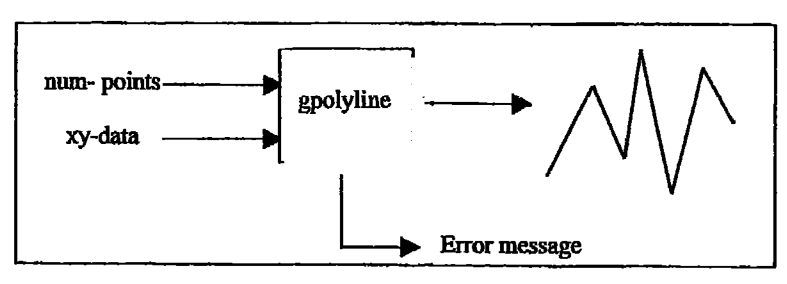
مرة أخرى ، نود أن نؤكد أن هذه القضايا ليست خاصة فقط بمنظومة GKS، ولكن نتوقع مواجهتها كلما استخدمنا قياسيات.

3.8.2 تناول الأخطاء (Error Handling)

عادة البرنامج الذي يستخدم حزمة برامج مستقلة عن الجهاز (Device- Independent Package) سوف ينفذ (run) ببطأ مقارنة مع مثيله الذي يكتب بطريقة

مثالية خاصة لجهاز واحد. تكون هنالك أسباب عدة لهذا البطء في الأداء: أحد هذه الأسباب الرئيسية هي كلفة تناول الأخطاء. إن ضرورة تتبع الأخطاء غالبا ما يتغاضى عنها أو ظاهره يسيء فهمها في البرمجيات المستقلة عن الجهاز.

لنأخذ بنظر الاعتبار دالة متعددة الخطوط (polyline). تتلقى هذه الدالة مدخلين (Two Inputs) وتقوم بإنتاج مخرجاتها على العارضة بشرط تكون قطع الخطوط واقعة ضمن النافذة وبالتالي لا نحتاج إلى التقليم. ما عدا هذه المعلومات، تكون الدالة عبارة عن صندوق أسود (Black Box) كما مبين في الشكل 31.2.



الشكل 31.2 متعدد خطوط كصندوق أسود

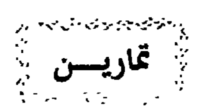
يستخدم مصطلح الصندوق الأسود من قبل المهندسين لوصف منظومة تحدد فقط بخواص مدخلاتها ومخرجاتها. نحن لا نعلم ما يجري داخل هذا الصندوق، بل نعلم فقط ما يلزم أن تكون مخرجاته لو تم تحديد مدخلاته. لكن بلغة برمجيات هذا يعني، أننا لا نعلم كيف تقوم حزمة برمجيات معينة بتنفيذ دالة متعددة الخطوط (polyline) أو أي دالة أخرى. وعلاوة على ذلك، أننا ليس ملزمين بمعرفة هذا إذا كانت البرمجيات في الحقيقة قياسية (Standard).

بدون آلية معينة لتتبع الخطأ، مع أن مفهوم الصندوق الأسود قد يتعرض إلى صعوبات خطيرة في كشف (تشخيص – Debugging) الخطأ في برامج المستفيد. افترض قمنا بكتابة برنامج مستفيد (User Program) وتم ربطه (Link) مع حزمة برمجيات للرسومات خاصة بنا، ومن ثم حدث عطل عند التشغيل. لنفترض نحن من المبرمجين الماهرين، نقوم بتشغيل كاشف الأخطاء (Debugger) ونستلم رسالة تقول " فيض في الدالة polyline عند العنوان 1234". قد تكون هذه المعلومة محدودة الاستعمال لنا، لأن

الدالة polyline تمثل الصندوق الأسود، ولا يمكننا التبصر في داخله لكي نتتبع الخطأ وإيجاد المشكلة. وبالتالي لا نستطيع ببساطة أن نعلم فيما إذا كان الخطأ في برنامجنا أم هناك خطأ في تنفيذ الدالة polyline أو نحن قد أخطئنا خلال التمهيد في البداية لدالة الرسومات. على سبيل المثال ، أي خطأ في مواصفات النافذة قد يسبب مشاكل عندما نحاول استخدام النافذة لتحديد فيما إذا قطع الخطوط المحددة بواسطة الدالة polyline ستظهر على عارضتنا. إذا حدث خطأ مثل هذا في المواصفات وكان مسبباً للمشكلة، وبدون آلية لتتبع الخطأ، قد يقوم الخطأ ببساطة بتوضيح نفسه من خلال الإشارة إلى مثل هذه الرسالة المبهمة "فيض في polyline" حتى لو كانت معلميات الدالة polyline صحيحة.

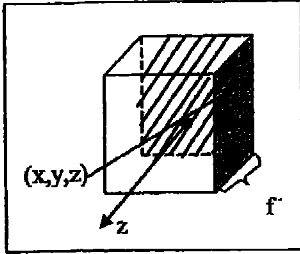
يكون الحل الجزئي لهذه الصعوبات هو ضرورة قيام كل دالة بتدقيق بيانات مدخلاتها وإعطاء رسالة أخطاء في حالة اكتشاف قيمة غير صحيحة.

من هنا، إذا حدث خطأ أثناء التنفيذ بسبب خطأ في مواصفات النافذة، تقوم دالة النافذة بإخراج رسالة خطأ عندما تدقق في بيانات المدخلات. إن الطرق المألوفة المستخدمة هي إما أن تقوم كل دالة بإعادة شفرة أو رقم الخطأ أو تتبع الأخطاء بواسطة تسجيلها على ملف أو على جهاز قياسي لتسجيل الأخطاء (Standard Error-Logging Device). في منظومتي PHIGS, GKS يتم تتبع الأخطاء بواسطة كل دالة تحمل رقم وحيد وكل خطأ محتمل حدوثه أيضاً يحمل رقم خاص به. على سبيل المثال، عند استدعاء الدالة polyline مع نقطة واحدة، يؤدي إلى توليد رسالة خطأ تقرأ بهذا الشكل "الخطأ 100 في الدالة 13". مع هذه المعلومات، تشخيص الأخطاء، بالرغم من أنها لا تزال تحتاج إلى مجهود شاق، لكنها تكون بمنتهى البساطة للمستفيدين.



1.2 إن عملية التزويم (تكبير بالتقريب Zooming) تجعل الشيء يظهر كبيراً أو صغيراً على العارضة. بين كيف يمكن تحقيق التزويم بواسطة تغيير إما حجم النافذة أو أبعاد الشيء (Scaling). هل يمكنك تشخيص الطريقة المستخدمة في التزويم بمراقبتك سلسله من الصور على العارضة ؟ هل إجابتك تعتمد فيما إذا كنا نعمل مع بعدين أو ثلاثة أبعاد.

2.2 يمكننا تركيب كامرة ذات فتحة إبرية (Pinhole Camera) وذلك بتكوين فتحة صغيرة في صندوق مكعب كما مبين في الشكل 32.2. البعد البؤري f للكاميرا عبارة عن المسافة بين الفتحة الإبرية والفيلم (Film) الذي قد تم وضعه في مؤخرة الصندوق. لنفرض أن مركز منظومة الإحداثيات هو عند الفتحة وأن اتجاه z يكون على امتداد المحور البصري (Optical Axis)، عمودياً على الفيلم في مؤخرة الصندوق. أين تظهر صورة نقطة عند (x,y,z) على الفيلم.



الشكل 32.2 كامرة ذات فتحة

3.2 افترض أنك تستخدم منظومة رسومات فيها تحديد الموقع المطلق (Absolute Positioning). صمم منظومة رسومات يكون فيها تحديد الموقع النسبي (Relative Positioning). أي بمعنى قم بكتابة إجرائين هما: move-to و line – to وذلك باستخدام الإجراء line فقط.

4.2 إننا نسمح لكل من النافذة وبوابة الرؤية بأن يكون لهما نسبة مربع أقصى (Aspect Ratio) مختلفين. والذي قد يسبب تشويه الشكل عندما يتم النقل إلى فضاء NDC. كيف يمكنك تغيير عملية النقل (Mapping) من أجل الحفاظ على الشكل بدون تشويه ؟ هل توجد هنالك سلبيات عند تنفيذ اقتراحاتك؟ لو كانت هناك أي سلبيات إشرحها.

5.2 كون منظومة تحديد موقع مشابه إلى LOGO من منظومة تحديد الموقع المطلق.

6.2 إن إحدى المعالم المهمة لمنظومة رسومات جيدة هو أن كل إجراء أو دالة تقوم بتدقيق بياناتها المدخلة من الأخطاء. أعد كتابة أجراء الصندوق لكي يتضمن تدقيق الأخطاء. أي نوع من الأخطاء ينبغي تدقيقه في إجراء الخط line.

7.2 يمكن وصف دائرة نصف قطرها r ومركزها عند نقطة الأصل بواسطة معادلتين معلمية (Parametric Equations) $x = r\cos\theta$ و $y = r\sin\theta$. يمكننا تصميم خوارزمية لتوليد دائرة وذلك بالبدء من نقطة تقع على الدائرة ولتكن (r,o) وبزيادة قيمة الزاوية θ بخطوات صغيرة. أشتق مثل هذه الخوارزمية. ومن ثم استخدم التقريب للجيب والجيب تمام (Sine and Cosine) للزوايا الصغيرة من أجل تسريع عملية تنفيذ البرنامج.

8.2 نستطيع وصف السطوح في ثلاثة أبعاد باستخدام معلميتين s,t.

$$x = f(s,t),$$
$$y = g(s,t),$$
$$z = h(s,t).$$

المطلوب توليد مجموعة معادلات معلمية للكرة . هل باستطاعتك إيجاد أكثر من تمثيل معلمي واحد لنفس الكرة؟

9.2 تصف المعادلات المعلمية التالية $x = xo + \alpha t$ و $y = yo + \beta t$ خط مستقيم عند النقطة (x_o, y_o) . أوجد علاقة بين الثوابت α و β مع ميل المستقيم.

10.2 عدة طرق لرسم الخط بواسطة المكونات المادية (Hardware) تستخدم الصيغة المعلمية كما في التمرين 9.2 لتوليد الخطوط. مع أن أجهزة العرض مثل، جهاز المسح العشوائي CRT، يقوم بتحريك الشعاع عند سرعة ثابتة v سم/ثانية. افترض عندنا قطعة خط تصل بين النقطتين (x_1,y_1) و (x_2,y_2). . كون صيغة معلمية بحيث عندما تزداد t من الصفر إلى الواحد، يتحرك الشعاع بنفس المعدل مستقلاً عن ميل قطعة الخط المستقيم.

11.2 غالباً هناك إرباك بين المنحنيات الفضائية (Spatial Curves) والمنحنيات بالفضاء المعلمي (Parameter Space). لنأخذ بنظر الاعتبار المنحني المعلمي

$$x(t) = t(1-t)$$
$$y(t) = t$$

ارسم هذه المنحنيات لمجموعة قيم t بين الصفر والواحد. ومن ثم ارسم هذه المنحنيات في إحداثيات (x,y) وذلك بتحديد قيم y,x من منحنيات x,t و y,t في الفضاء المعلمي.

12.2 أعد حل تمرين 11.2 مع المنحني المعلمي:

$$x(t) = t(1-t)$$
$$y(t) = t(1-t).$$

13.2 لنأخذ منحني ثلاثية الأبعاد معرف بواسطة الدوال الثلاثة الآتية:

$$x(t) = \sin t$$
$$y(t) = \cos t$$

$$z(t) = t$$

لقيم $t \geq o$. خطط رسم ثلاثة منحنيات معلمية والمنحني الناتج في فضاء (x,y,z).

14.2 لديك نقطتين p1,po في البعد الثنائي، اكتب معادلة قطعة الخط الواصلة بينهما في صيغة

$$p(\alpha) = p_0 + \alpha d,$$

حيث أن d وحدة طول (Unit Length). أوجد قيمة α للحصول على p_1.

15.2 اقترح نوعين مختلفين من البيانات للنافذة (Window) ونوعين لبوابة الرؤية (Viewport).

16.2 لأي نوع من الأخطاء ينبغي على الإجرائين النافذة وبوابة الرؤية التأكد منها عند تنفيذهما وظائف النافذة وبوابة الرؤية.

الفصل الثالث

رسومات ثنائية البعد (Two – Dimensional Graphics)

الفصل الثالث

رسومات ثنائية البعد *(Two – Dimensional Graphics)*

مقدمة:

في هذا الفصل، سنرى بناء أول مجموعة برامج كاملة للرسومات. سوف تستخدم هذه البرامج دوال من مكتبة الرسومات (Graphics Library) للتزود بما هو ضروري من واجهة بينية (Interface) مع منظومة الرسومات. وبعد إلقاء نظرة عامة على أنواع الدوال المتوقع إيجادها في أي منظومة حديثة، سوف نوجه اهتمامنا نحو برامج الإخراج فقط (Output-Only) ومن النوع المستخدم في إنتاج مخططات بيانية(Graphs) نموذجية.

وفي الفصل القادم، سوف نناقش برامج تتضمن التفاعل مع المستفيد (User Interaction).

سوف نستخدم منظومة نواة الرسومات GKS لتطوير برامجنا. مع أننا سنحاول استخدام منظومة GKS بصورة نظامية، وكذلك سوف نحاول تفادي التفاصيل الخاصة بهذه المنظومة. برامجنا يجب أن تكون لها قابلية التنقل إلى منظومات رسومات أخرى مع حد أدنى من التغييرات. سنقوم بتطوير إجراء بسيط لراسم بيانات ذاتي التدرج (self-Scaling Plotter Procedure) كمثال لبرمجة رسومات مستقلة عن الجهاز. أيضاً سنتعرف على ملفات ملحقة (Metafiles) التي سوف تسمح لنا بنقل معلومات عن الرسومات بين مختلف البرامج والمنظومات.

1.3 قياسيات رسومات مستقلة عن الجهاز

(Device-Independent Graphics Standards)

قادتنا الفعاليات المتنوعة إلى تحديث لغات برمجة رسومات قياسية (Standard Graphics-Programming Languages). كانت الدوافع هنا مشابهة لتلك التي أدت إلى

تطوير لغات برمجة قياسية مثل لغة Pascal ولغة C ولغة فورتران. قد تكون رغبة المبرمجين في كتابة برامجهم على منظومة معينة وهم على علم إن باستطاعتهم نقلها إلى منظومة أخرى وبدون معرفة فعلية بتفاصيل المكونات المادية لأي من المنظومتين. لا تستوجب حاجتهم في معرفة تفاصيل العمليات المحددة للمنظومة، كعمليات الإدخال والإخراج، عند كتابة معظم البرامج التطبيقية. مثال ذلك، في أكثر لغات البرمجة عالية المستوى (High-level) تنجز عملية الإدخال والإخراج من خلال وحدات منطقية قياسية (Standard Logical Units) التي يتم تحديدها بأرقام أو رموز كمثل Printf في لغة C أو (*,*) Write في لغة فورتران . إن مثل هذه البرامج تدعى مستقلة عن الجهاز (Device Independent).

توجد هنالك فوائد أخرى للقياسات. أن التطورات الحديثة التي طرأت على لغات البرمجة والزيادة المطردة في البرامج المنتجة أدت إلى زيادة عدد المستفيدين اللذين أصبحوا متآلفين مع المنظومة، حيث ظهرت طرق قياسية لإنجاز مهمات كتناول الأخطاء (Error Handling). وكلما نمى جمهور المستفيدين أصبح توفر مكونات مادية وبرمجيات أفضل وبأقل كلفة. إن المعرفة المكتسبة في تطبيق واحد يمكن بسهولة الاستفادة منه في التطبيق التالي. أما من وجهة نظر المكونات المادية، هناك عدة فوائد لبرمجيات الواجهة البينية القياسية (Standard Software Interface). في منظومة الرسومات، على سبيل المثال، مصمم المكونات المادية يعلم مسبقاً ما هي الكيانات الأولية للإخراج (Output Primitives) المطلوبة من المكونات المادية توليدها.

سوف نستخدم منظومة GKS في تطوير برامجنا وكذلك في توضيح القدرات المتاحة لحزمة برمجيات الرسومات الحديثة معاً. لقد تم تطوير منظومة GKS متذكرين هذه الأفكار. إن منظومة GKS هي حصيلة ذروة عمل عدد من الكوادر المتخصصة آخذين بنظر الاعتبار لغات رسومات مقترحة أخرى كمنظومة CORE. أما من وجهة نظر المستفيد فمنظومة GKS عبارة عن مجموعة دوال يمكن استدعائها من خلال البرنامج التطبيقي . يستطيع المستفيد استخدام هذه الدوال في تطوير تطبيقه أو ينتفع منها كأساس في تصميم مجموعة دوال أخرى لشريحة من المستفيدين المتخصصين، كمصمموا الدوائر الكهربائية أو المعماريون.

يمكنك أن تجد نبذة تأريخية عن منظومة GKS وكيف تم اختيارها موثقاً في مصدر آخر مثل [End 84]. ومن الجدير بالاهتمام أن تدرك الصعوبات التي تتضمنها تلك المنظومة والقضايا التي برزت والقياسيات المتضاربة التي أخذت بنظر الاعتبار من قبل منظمات القياسيات (Standards Organizations). منذ انبثاق منظومة GKS من العملية كمنظومة قياسية، مع ذلك سنواصل من تلك النقطة. هنالك قياسيات أخرى للمبرمج مثال ذلك، قياسيات منظومة PHIGS التي لها هيكلية مشتركة مع GKS، فالبرامج البسيطة في أحد القياسيات يكون في الواقع مماثل لبرامج بسيطة في قياسيات أخرى.

ببساطة لو كان تصورنا لمنظومة GKS هي عبارة عن مكتبة رسومات (Graphics Library)، لفقدنا كثير من الأفكار الأساسية المتضمنة في المنظومة. بصورة خاصة هنا قد تكون المقارنة مع لغات البرمجة واردة. إن الكيفية التي يتم فيها حل مشكلة معينة قد تختلف كثيراً معتمدة على لغة البرمجة المستخدمة. حيث يكون الاختلاف في ما نستطيع عمله وكيف يتم تنفيذه في لغة فورتران أو لغة LISP. إن التطور الحاصل في لغات البرمجة الحديثة ناشئ بصورة كبيرة عن حاجة المبرمجين والمتخصصين في علوم الحاسبات إلى وسائل أكثر فعالية للعمل معها. كما هو الحال مع منظومات الرسومات بالحاسوب. إن منظومات PHIGS, GKS تتضمن العديد من الأفكار الأساسية حول كيفية تطوير برامج الرسومات. لقد سبق وإن لاحظنا بعض هذه الأفكار مثل المناظرة بآلة التصوير الاصطناعية وغيرها من الأفكار التي سنقوم بتطويرها أثناء دراستنا لهذه المنظومات.

إن المعالم الرئيسية التي توفرها حزم برامج الرسومات تتضمن:

1- محطات عمل منطقية متعددة.
2- تجزئة وتحويل الصور.
3- سيطرة متطورة لعارضات محطات العمل وأجهزة الإدخال.
4- تخزين واسترجاع معلومات الرسومات.
5- تحكم المستفيد في تناول الأخطاء.

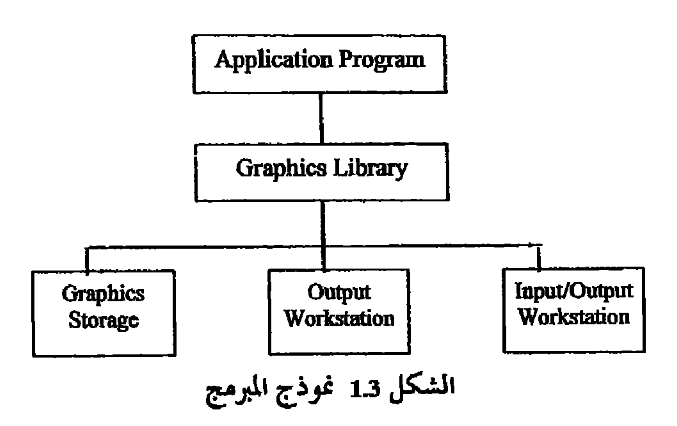
في هذا الفصل، ستتعرف على المستويات الدنيا (Lowest Level) لبرمجة الرسومات. حيث تقوم برامجنا بإنتاج مخرجات معتمدة على استخدام مباشر للكيانات الأولية المتوفرة

من قبل المنظومة. عندما يكون الإدخال غير ضروري، العروض تولد مرة واحدة فقـــط. تستخدم مثل هذه البرامج في رسم البيانات حيث تحاكي دوال الرسومات المطلوبة الراسم القلمي. حزم برامج الرسومات البسيطة التي تعتمد على هذا النموذج غالباً مـــا تدعـــى أدوات رسومات (Graphics Toolkits).

أثناء عملية تطوير هذا النموذج، سوف نتآلف مع الكيانـــات الأوليـــة للإخـــراج، صفاتها المميزة (Attributes) والمشاهدة ثنائية البعد. نحن لا ندعي بالكمال، مع أننا نقـــوم بتطوير برامج صحيحة. يكون هدفنا هو تعلم كيفية برمجة تطبيقات رســـومات بســـيطة مستخدمين منظومة GKS كأداة.

2.3 نموذج المبرمج (The Programmer's Model)

عندما نستخدم منظومة رسومات، غالباً ما يستخدم المبرمج التطبيقي نموذج مفـــاهيمي (Conceptual Model) لمعمارية منظومة الرسومات كما هو مبين في الشكل 1.3.



الشكل 1.3 نموذج المبرمج

هذا النموذج يتضمن:

1- القدرة على احتواء أجهزة إدخال وإخراج معاً.
2- القدرة على احتواء أجهزة متعددة تحت سيطرة مستفيد واحد.
3- القدرة على تخزين ونقل معلومات الرسومات.

لا يتضمن النموذج أية تفاصيل تتعلق بالتنفيذ. بالنسبة لهذه التفاصيل تكـــون غـــير ضرورية، لأن المبرمج يستطيع الوصول إلى هذه القدرات من خلال نداءات لإجـــراءات

(Procedure Calls) منظومة الرسومات والتي تكون مستقلة عن الجهاز. هكـــذا تكـــون محطات العمل بالنسبة للمبرمج التطبيقي كوحدات منطقية، بدلاً من وحدات حقيقيـــة أو فعلية (Physical).

1.2.3 محطات عمل منطقية وحقيقية : (Logical and Physical Workstations)

عندما نجلس أمام محطة طرفية للرسومات أو حاسوب شـــخصي أو محطـــة عمـــل للرسومات المتطورة، نحن نرى جهاز حقيقي يحتوي على عارضة CRT وجهاز واحـــد أو أكثر من أجهزة إدخال كلوحة المفاتيح وفأر. الأشـــكال التكوينيـــة (Configurations) المختلفة لمحطة العمل لها خواص مادية أو حقيقية مختلفة. لربما أحدها تمتلك عارضة أحاديـــة اللون ذات دقة عالية والأخرى لها عارضة ملونة ذات دقة واطئـــة (Low-Resolution). مع ذلك نرغب بتطوير برمجيات مستقلة عن الجهاز. لذا نرغب أن يشتغل نفس برنـــامج المستفيد على كلا هاتين العارضتين، وكذلك لا نريد القيام مثلاً بتغيره إذا كانت منظومـــة معينة تحتوي على لوحة بيانات (Data Tablet) بدلاً من الفـــأر. إن إحـــدى الأســـس للرسومات الحديثة بالحاسوب هو التعامل مع أجهزة منطقية (Logical Devices) بـــدلاً من أجهزة حقيقية (Physical Device) في برنامج المستفيد. لذا تسمح لنا محطات العمـــل المنطقية (Logical Workstation) تطوير برامج تطبيقية مستخدمين منظومة الرســـومات بدون الحاجة لمعرفة الخواص المادية للأجهزة المعينة كلاً على الانفراد.

يتميز الجهاز المنطقي بوظائفه التي يؤديها بدلاً من صفاته المادية أو الحقيقية. إن هـــذا المفهوم ينبغي أن يكون مألوفاً من لغات البرمجة القياسية. مثـــال ذلـــك، دوال الإدخـــال والإخراج في لغة C هي scanf, printf, write, read كل هذه الدوال تشير إلى أجـــهزة منطقية، حيث إن أول اثنين تتعلق بالبيانات الثنائية والزوج الثـــاني يتعلـــق بالحرفيـــات (Characters). لذا عند كتابة برامج، علينا أن لا نهتم فيما إذا كانت المدخلات آتية مـــن محطة طرفية أو من ملف على قرص. إن إمكانية إعادة توجيه المدخلات والمخرجات مـــن محطة طرفية إلى ملفات في عديد من منظومات التشـــغيل (Operating Systems) أيضـــاً يوضح ذلك، حيث ضمن برنامج المستفيد نقوم بالتعامل مع أجهزة منطقية وليس مـــع أجهزة مادية.

في الرسومات ، استخدام محطات عمل منطقية نوعا تكون أكثر تعقيداً، لأننا نرغب في توفير نوع من القدرات الواسعة للمستفيد. بعض محطات العمل لها القابلية على إنتاج مخرجات فقط، في حين الأخرى تنتج مخرجات إضافة إلى تجهيز مدخلات لبرامجنا. لربما أنواع أخرى من محطات العمل قد توفر إمكانية خزن للرسومات وقواعد البيانات (Databases). من الآن سنعتبر لدينا ثلاثة أنواع من محطات العمل المنطقية وهي :

1- الإخراج 2- الإدخال 3- الإدخال والإخراج معاً.

قد يتحكم برنامج المستفيد بمحطات عمل منطقية متعددة، سواء أكانت محطات العمل المتعددة هذه لها ما يقابلها من أجهزة حقيقية منفصلة أم لا.

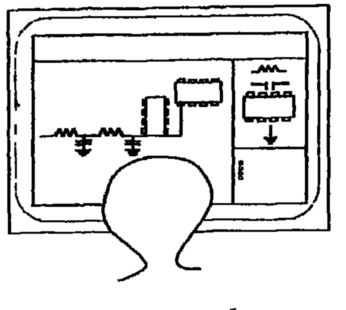
إن محطة عمل للإخراج تحتوي على سطح عرض منطقي واحد الذي تعرض عليه الرسومات. بإمكان سطح العرض إظهار أية كيانات أولية تنتجها المنظومة كمتعددة خطوط ونصوص. أما محطة العمل للإدخال قد تحتوي على جهاز واحد أو أكثر من أجهزة إدخال منطقية: قد تقوم هذه الأجهزة بتجهيز برنامج التطبيق بمدخلات الرسومات. في الفصل القادم سنرى كيف أن مدخلات الرسومات يمكن أن تأخذ عدة أشكال. تحتوي محطات العمل للإدخال والإخراج على سطح عرض منطقي واحد مع جهاز واحد أو أكثر من أجهزة إدخال منطقية.

إن القرار المتعلق بكيفية تنظيم محطات العمل المنطقية غالباً ما يترك إلى مبرمج التطبيقات. مثال ذلك، افترض لو أردنا تصميم برنامج رسم متفاعل يتم تنفيذه على حاسبة شخصية . أية حاسبة شخصية نموذجية ستحتوي على عارضة CRT ولوحة مفاتيح والفأر وراسم بيانات كما مبين في الشكل 2.3.

الشكل 2.3 الشكل التكويني لحاسوب شخصي

نظراً لأننا نرغب في تصميم برنامج مع واجهة بينية تساعد المستفيد، لذا ستعرض الشاشة لنا عدد من قوائم الاختيار (Menus) وشواخص أو دلالات (Icons) كما هو مبين في الشكل 3.3.

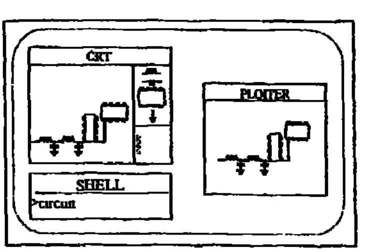
إن معظم الوقت يقضيه المستفيد مع هذا البرنامج هو التفاعل مع الشاشة، وأحياناً يستخدم الفأر (مثل رسم المنحنيات) وأحياناً أخرى مستخدماً لوحة المفاتيح (مثل إدخال صفوف من الرموز للهوامش). عندما نريد الحصول على نسخة مطبوعة (Hardcopy)يتم إرسال جزء من المعروض على الشاشة إلى راسم البيانات (تهمل الشواخص وقوائم الاختيار).



الشكل 3 3 عارضة المستفيد

في هذا المثال، ينبغي أن يكون لدينا على الأقل محطتين عمل منطقية، نظراً لوجود عارضتين حقيقتين يتعين السيطرة عليهما. يسمح هذا التقسيم المنطقي للمبرمج السيطرة على عارضتين بصورة مستقلة، أنها خاصية مميزة ومطلوبة جداً، نظراً لرغبتنا في إرسال الكيانات الأولية للرسومات إلى العارضة CRT قد تختلف عن تلك التي نريد إرسالها إلى راسم البيانات (Plotter).

نستطيع التعامل مع جهازي الإدخال بعدة طرق. يمكن إشراك كلا الجهازين مع محطة عمل إدخال منطقية واحدة. أو يمكننا إشراك كلا الجهازين مع إحدى العارضتين، هكذا إذن يتم تحديث محطة عمل منطقية للإدخال والإخراج. أيهما يتم استخدامه فعلاً سيكون تحت تصرف المبرمج، نستطيع الاختيار من أجل جعل البرمجة سهلة. مع أن ، قد يظهر هناك وجود تناظر أحادي (One-to -One Correspondence) بين محطات العمل المنطقية والحقيقية، ولكن هذا ليس بالضرورة أن يكون موجود. على سبيل المثال، لنفترض تم تطوير برنامجنا للرسم (Painting Program) على محطة عمل ذات فعالية عالية، حيث تتصف عارضتها بدرجة وضوح عالية (High-Resolution) والقدرة على تعدد المعالجات (Multiprocessing). على منظومة مثل هذه ، نستطيع عرض كلتا المخرجات المنطقية على نفس العارضة الحقيقية CRT (كما في الشكل 4.3).



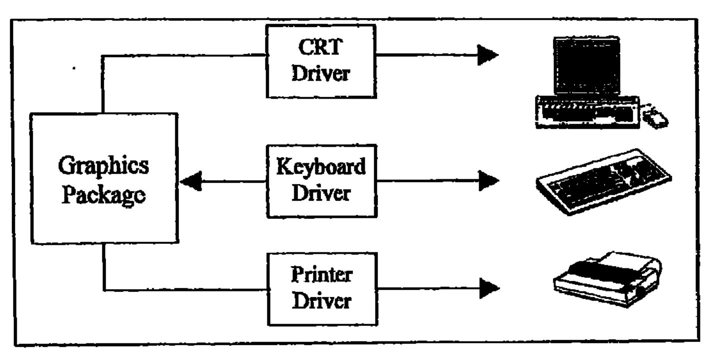
الشكل 4.3 عارضة على محطة عمل تطويرية

وذلك بتخصيص أجزاء مختلفة من السطح الحقيقي للعارضة CRT إلى محطتين عمل منطقية. بسبب استطاعتنا إعادة تخصيص الأجهزة المنطقية إلى أجهزة حقيقية – بواسطة آلية بسيطة سنشرحها لاحقاً – يتم تنفيذ نفس البرمجيات بدون تعديل على كل من الحاسبة الشخصية ومنظومة التطوير (Development-System).

2.2.3 الاتصال مع المكونات المادية (Communicating With The Hardware)

يقوم المستفيد بالاتصال مع الأجهزة الحقيقية بصورة غير مباشرة فقط، ومن خلال نداءات إلى إجراءات الرسومات . لذا تبدوا محطة العمل الحقيقية كجهاز منطقي مسيطر عليه بواسطة هذه النداءات. نظراً لكون المستفيد يكتب البرنامج بلغة برمجة معينة، لذا يتم استحضار هذه النداءات من خلال مجموعة أسماء قياسية ومصطلحات ندائية تعرف بالترابط اللغوي (Language Binding). عادة تكون هذه الإجراءات موجودة في مكتبة يستخرج منها الإجراءات الضرورية أثناء الترابط مع برنامج المستفيد. يكون هذا المستوى تحت مستوى برنامج المستفيد كما في الشكل 1.3، وغالباً يكون هذا كل ما يحتاجه المستفيد أن يعلمه حول الهيكل الداخلي لمنظومة الرسومات (Graphics System).

بالإمكان تصوير التفاعل بين المكونات المادية والبرمجيات كما مبين في الشكل 5.3.



الشكل 5.3 مكونات مادية – برمجيات الواجهة البينية

توفر وحدات إدارة أجهزة (Device Drivers) الواجهة البينية بين البرمجيات والمكونات المادية. يتم السيطرة على كل جهاز حقيقي بواسطة وحدة إدارة جهازه التي تكون أما جزء من البرمجيات أو محتوياً في المكونات المادية.

بالنسبة لمنظومة الرسومات ، يتم إنتاج المخرجات من حزمة برامج الرسومات بهيئة إحداثيات معيارية للجهاز NDC. ومن ثم يتم تحويل هذه البيانات إلى أوامر وبيانات دقيقة للأجهزة الحقيقية ممثلة في وحدات إدارتها (Drivers). يتم تناول المدخلات بعكس هذه الطريقة . حيث يتم تحويل بيانات من أجهزة الإدخال إلى هيئة إحداثيات في منظومة NDC بواسطة وحدات إدارة الجهاز، ومن ثم ترسل إلى إجراءات الرسومات . في تطبيق جيد التصميم، تكون وحدات إدارة الجهاز، الجزء الوحيد من المنظومة معتمدة على الجهاز. لذا عند إضافة جهاز إخراج أو إدخال جديد للمنظومة يتطلب فقط إضافة وحدة إدارة جهاز مناسبة إلى المنظومة .

3.2.3 قضايا تنفيذية: (Implementation Issues)

إن القضايا والتساؤلات التي تحيط بالتنفيذ عديدة ومختلفة. وهي تتضمن ما يلي:

1- كيف يتم إنتاج كيانات أولية على عارضة حقيقية.

2- كيف تعمل أجهزة الإدخال المختلفة.

3- كيف يتم كتابة برمجيات وحدات إدارة الجهاز.

4- كيف تم تنظيم وكتابة حزمة برامج الرسومات مثل GKS.

عند هذه النقطة، نحن ندعي بأننا نقوم في تطوير تطبيقات رسومات مستقلة عن الجهاز والذي يقضي بعدم الحاجة إلى الاهتمام بقضايا التنفيذ. ولكن علينا أن نميز بأن هناك يوجد فرق بين العمل مع محطة طرفية مرتبطة بحاسوب حاضن (Host Computer) عبر أداة بينية متتابعة (Serial Interface) والعمل مع محطة عمل مسح شبكي ملونة (Color Raster Workstation).

نحن سنتبنى أسلوب وسطي. عندما نقوم بمناقشة برامج تطبيقية مكتوبة في منظومة GKS، سنهمل معظم الأمور المتعلقة بالتنفيذ. ولكن كلما انتقلنا إلى رسومات محسنة وأكثر تطوراً (في الفصل الخامس) سنجد لم يعد يمكننا إهمال قضايا التنفيذ كلياً. في الفصل السادس والسابع، سنأخذ بنظر الاعتبار التنفيذ بدقة أكثر، وسنناقش بالتفصيل قضايا تشمل وحدات إدارة الجهاز، والواجهات البينية للأجهزة (Device Interface) واستخدام مكونات مادية حقيقية.

3.3 دوال الرسومات(Graphics Functions)

نموذجنا لمبرمج التطبيقات استخدم منظومة رســـــومات يلـــزم المـــبرمج الوصـــول للرسومات من خلال دوال تتضمنها المكتبة. قد تحتوي حزمة كاملة للرســـــومات علـــى مئات الدوال. وقد لا يحتاج تطبيق معين جميع هذه الدوال المختلفة . فعلى سبيل المثـــال، النشر بواسطة حاسوب منضدي غالباً ما يأخذ جميع مدخلاته من ملف نصوص بســـيط ولكن يتطلب إلى مخرجات عالية النوعية (High- Quality Output). أمـــا في تطبيقـــات أخرى، مثال ذلك، محاكيات الطيران (Flight Simulators)، قد لا تحتاج إلى مخرجـــات عالية النوعية، ولكن بالتأكيد يحتاج إلى مدخلات عالية النوعية. إذا تطلـــب مـــن جميـــع التطبيقات أن تدرك جميع الدوال، هذا قد يؤدي إلى تطبيقات بالغة التعقيد وبطيئة.

بدلاً من التزام جميع المنظومات تنفيذ كل دوال منظومـــة GKS ، تســـتطيع تمييـــز مستويات مختلفة للتنفيذ. على سبيل المثال، توجد هناك ثلاثة مستويات مختلفة للإخـــراج (2,1, 0) وثلاثة مستويات للإدخال (a,b,c). هذه المستويات بدورها تقوم بتعريف تســـعة مستويات تنفيذ أو تسعة فئات من دوال منظومة GKS البالغة 200 دالة تقريباً. ســـوف نقوم بتقسيم هذه الدوال إلى ثمانية صنوف وهي: الإخراج، التحكم (السيطرة)، الصفات، التحويل، الإدخال، التجزئة، الملفات الملحقة، التساؤل أو الاستعلام (Inquiry).

1.3.3 دوال الإخراج (Output Functions)

تحتوي منظومة الرسومات على مجموعة كيانات أولية للرسومات التي يمكن عرضـــها على أجهزة إخراجها. تكون هذه الكيانات الأولية عبارة عن كيانات قائمة بذاتها عنـــد أدنى مستوى (Lowest-Level)، التي يستخدمها برنامج المستفيد. تقوم دوال الإخـــراج بتعريف "ماذا" ستكون مخرجات منظومة الرسومات.

لدى منظومة GKS خمسة كيانات أولية وهي كما يلي:

1- متعدد الخطوط Polylines

2- النص Text

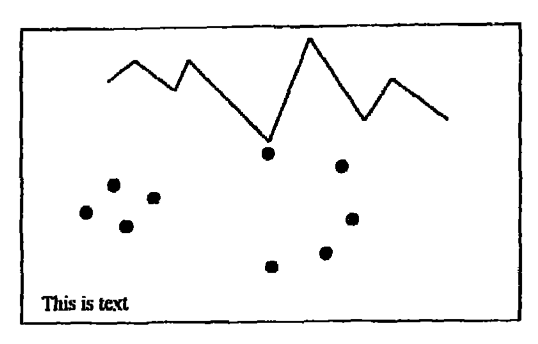
3- متعدد العلامات Polymarkers

4- مساحة ملء Fill Areas

5- صفوف خلية Cell Arrays

المبينة في الشكل 6.3. لكل كيان أولي هنالك دالة إخراج ، مثل

gpolyline (num-points, array-of-points)



الشكل 6.3 كيانات أولية لمنظومة GKS

هذه المجموعة من الكيانات إلى حد ما صغيرة. ومن الطبيعي نستطيع بناء مزيداً من كيانات مركبة للرسومات وذلك بكتابة دوال خاصة بنا مستخدمين هذه الكيانات الأولية. وكذلك يمكننا إضافة مزيد من الكيانات الأولية. لربما الدوال المرشحة، في البعد الثنائي، هي منحنيات عديدة الحدود (Polynomial Curves)، مربعات، مستطيلات، دوائر والقطع الناقص (اهليج-Ellipse).

إن اختيار الكيانات الأولية الخمسة المستخدمة في منظومة GKS اعتمدت على عاملين أولهما: من المتوقع بسهولة إنتاج هذه المجموعة فعليا على جميع المكونات المادية، وثانياً كل كيان أولي يوفر إمكانية لا يمكن الوصول إليها من الآخر.

2.3.3 دوال التحكم أو السيطرة: (Control Functions)

يكون من الضروري توفر عدد من دوال التحكم في منظومة الرسومات للسيطرة على عارضات محطة العمل، وأجهزة الإدخال وحالة المنظومة. أمثلة على ذلك:

1- ضرورة تمهيد المنظومة (Initialized System).

2- إعداد جداول الحالة (State Tables).

3- فتح ملفات الأخطاء.

4- تثبيت القيم الابتدائية أو الافتراضية (Default Values) للمعلميات.

ينبغي إتخاذ قرارات مثل كيف ومتى نقوم بتحديث عارضات محطات العمل. تكـون إحدى الخواص المهمة لمنظومة جيدة التصميم هي المرونة التي توفرها للمستفيد من خـلال دوال تحكمها.

كثير من دوال التحكم، كتلك المتضمنة في تمهيد وإنهاء برنامج الرسومات، في الواقع تكون نفسها في جميع البرامج. هكذا، بالرغم من أننا سنقوم باستعراضها لكونها ضروريـة بعدئذ يمكن إخفائها ضمن بعض دوال مكتوبة تستخدم في برامج لاحقة.

3.3.3 الصفات المميزة (Attributes):

يمكن أن تعرض الكيانات الأولية بعدة طرق. فمثلا يمكن أن يكون الخط غليـط أو رفيع، مقطع أو متصل، أحمر أو أخضر. وكذلك يمكن أن تكون الرموز في حجوم مختلفـة أو تعرض أما أفقياً أو عمودياً، أو قد تكون لها ألوان مختلفـة. دوال الصفـات تعـرف "كيف" يتم عرض الكيانات الأولية.

4.3.3 دوال الرؤية والتحويل (Viewing And Transformation Functions)

تسمح دوال الرؤية للمشاهد أن يحدد معلميات الرؤية (Viewing Parameters) أما بالنسبة لمنظومات الرسومات هي أين نضع الصورة . تقوم هذه الدوال بتحديد أي جـزء من عالم الإحداثيات سيظهر على العارضة ومكان ظهور الكيانات الأولية عليها. اذن هذه الدوال تشتمل على دوال النافذة وبوابة الرؤيـة (Window And Viewport)والتقليـم (Clipping) لمنظومة الرسومات. نظراً لكون هذه الدوال تحدد التنقـلات (Mappings) المختلفة بين منظومات الإحداثيات – على سبيل المثال، من WC إلى NDC ومـن NDC إلى DC – تقوم هذه الدوال بتحديد بعض التحويلات التي يتم تأديتها من قبل المنظومة.

توفر منظومة GKS مستويين من الرؤية:

1. تحديد أي جزء من منظومة إحداثيات WC يتم نقله إلى بوابة الرؤية في فضـاء منظومة إحداثيات NDC. يعـرف التحويـل النـاتج بـالتحويل المعيـاري (Normalization Transformation) . يمكن أن تتواجد عدة تحويلات معياريـة في آن واحد والتي سوف نراها لاحقاً حيث توفر للمستفيد وسيلة مفيدة.

2. يكون المستوى الثاني للرؤية هو مستوى محطة العمل، حيث يسمح للمستفيد بعرض أجزاء مختلفة من فضاء NDC على محطات عمل مختلفة. هذا التحويل يعرف بتحويل محطة العمل (Workstation Transformation)، حيث يقوم بنقل القيم في فضاء NDC إلى منظومة إحداثيات الجهاز (DC). يوجد لكل جهاز إخراج تحويل محطة عمل.

في حين عمليات الرؤية تقوم بتعريف تحويلات بين منظومات الإحداثيات، غالباً ما نرغب إجراء تحويلات ضمن منظومة إحداثيات واحدة، ومثال ذلك، عندما نقوم بتدوير خط أو تحريك شيء منظور من موقع معين إلى آخر، في الواقع نحن نقوم بتأدية عملية تحويل. عادة توفر منظومة الرسومات بعض دوال خدمية أو منفعية (Utility Functions) لهذا الغرض.

تكون دوال التحكم، الإخراج والتحويل ضرورية في أي برنامج رسومات. أما الصنوف الأخرى (الإدخال، التجزئة، الملف الملحق والاستعلام) تعطي المنظومة قابلية في توفير وسيلة للتفاعل وقدرات في معالجة الصور التي ترافق الرسومات الحديثة بالحاسوب.

5.3.3 دوال الإدخال (Input Functions)

من وجهة نظر المبرمج، يجب أن تكون دوال الإدخال مستقلة عن الجهاز، بالضبط كما هو الحال مع دوال الإخراج. تكون مدخلات الرسومات متنوعة كثيراً بالمقارنة مع مدخلات البرامج القياسية، وذلك لأنها تأتي في صيغ متعددة. في برنامج رسم البيانات (Plotting Program) الذي سوف نقوم بتطويره لاحقاً في هذا الفصل، قد نطلب من المستفيد إدخال عنوان الرسم البياني وعلامات المحاور، كل هذه المدخلات عبارة عن نصوص (Text Input). لو أراد المستفيد أن يدخل البيانات لرسم المخطط البياني، فقد تأخذ تلك المدخلات صيغة مواقع على الشاشة. في تطبيق التصميم المعزز بالحاسوب (CAD)، غالباً ما نرغب في تحريك أشياء منظورة حول الشاشة. كل واحد من هذه المدخلات تمثل صيغ مختلفة من المدخلات المنطقية (Logical Input)، وأنها توفر صيغ متنوعة تماماً من المعلومات إلى برنامج المستفيد، بصرف النظر عن الجهاز الحقيقي المستخدم لإدخال هذه المعلومات.

تقوم منظومة GKS بتعريف ستة صنوف من المدخلات المنطقية وهي:

1. محدد المواقع (Locator)
2. صف من الرموز (String)
3. التقاط (Pick)
4. اختيار (Choice)
5. ضربة (Stroke)
6. مقيم أو مخمن (Valuator)

يمكن أن يحدث الإدخال في أكثر من نمط واحداً أيضاً. لنأخذ على سبيل المثال، محاكي الطيران، الذي يجب أن يستجيب مع الحالة الحاضرة لأجهزة إدخاله في جميع الأوقات فوراً. في حالات أخرى، كما في معظم تطبيقات CAD، هنا قد يحتاج المستفيد إلى وقت لتحديد وضع الجهاز أو أن يدخل رموز من لوحة المفاتيح. في هذه التطبيقات، ينبغي على المستفيد أن يصدر إشارة قبل بدء معالجة المدخلات. تسمح المنظومتين PHIGS, GKS لكلا النوعين من المدخلات وذلك بالسماح للمبرمج في تحديد نمط الإدخال (Mode Of Input). بالإضافة إلى ذلك توجد هنالك عادة بعض أنواع الأوامر التمهيدية للتحكم بمحطات العمل للإدخال. كل هذه القضايا سوف نبحثها في الفصل الرابع.

6.3.3 دوال التجزئة (Segmentation Functions)

بالرغم من أن الكيانات الأولية تسمح للمستفيد أن يقوم بوصف صورة بالغة التعقيد وشاملة، لكن المجموعات البسيطة من الكيانات الأولية المتوفرة على معظم منظومات الرسومات لا تجعل هذه المهمة سهلة. تعمل معظم التطبيقات الحقيقية مع كيانات أكثر تعقيداً بكثير من متعدد خطوط منفرد (Single Polyline). على سبيل المثال، في تطبيق تصميم الدائرة الكهربائية، حيث تستخدم رموز للمقاومات والمكثفات والدوائر المتكاملة بصورة متكررة، وقد لا يرغب المستفيد إعادة تعريف هذه العناصر أو الرموز بصورة متكررة. عند مستويات عليا في هذا المثال. قد يرغب المستفيد التعامل مع دوائر ثانوية (Subcircuits) كاملة. هنا دوال التجزئة (Scgmentation Functions) تمكن المستفيد من تكوين ومعالجة مجموعات من كيانات أولية مميزة تدعى أجزاء أو قطع (Segments).

حالما يتم تعريف قطعة، يكون بمقدورنا تحويل وتغيير الأشياء المنظورة بدون الاهتمام حول المركبات المستقلة عند المستوى الأدنى. هكذا ، سنكون قادرين على إعادة تقييس (Rescale) هيكل كامل معرف من قبل المستفيد، مثال ذلك، مخطط بيانات أو دائرة كهربائية مصممة بواسطة أمر أو إجراء واحد . من ناحية الإدخال ، نستطيع اختيار شيء منظور على شاشة العرض بالتأشير على أي جزء منه ومن ثم إيصال هذه المعلومة بالعودة إلى برنامج المستفيد.

7.3.3 ملفات ملحقة (Metafiles)

توفر دوال الملف الملحق (Metafile Functions) آلية إمساك واستعادة الصورة. سيكون بمقدورنا الاحتفاظ بالصورة (أو جزء من الصورة) حيث يكون من الممكن استعادة عناصر الصورة المنفردة بواسطة برنامج تطبيقي. بالإضافة إلى ذلك أن سهولة إمكانية خزن صورة في ملف، أصبحت بنفس سهولة إمكانية إنتاج صورة على محطة عمل – في الحقيقة تكون المفاهيمية الآلية (Conceptual Mechanism) لهما مماثلة. وهكذا بدلاً من الحاجة إلى نقل أو خزن صورة إما بشكل نسخة مطبوعة (Hardcopy)أو بطريقة غير مباشرة وذلك بواسطة البرنامج الذي أنتج الصورة، أصبح يمكننا القول أن الملف الملحق يوفر لنا خيار ثالث.

8.3.3 دوال استعلامية (استفسارية) (Inquiry Functions)

مع أننا عادة نرغب كتابة برامج مستقلة عن الجهاز، لكن أحياناً نريد معرفة بعض الأشياء حول المكونات المادية أو ما يدور داخل منظومة الرسومات. فمثلاً، بالرغم أننا نستطيع تحديد لون اختياري للخط أو مساحة ملء (Fill Area)، قد نرغب في معرفة فيما إذا كنا نعمل مع منظومة ملونة. إذا ظهر لدينا أن المنظومة أحادية اللون (Monochromatic – مثال راسم بيانات ليزرية Laser Plotter)، آنذاك لربما نرغب تعويض الخطوط الخضراء بخطوط متقطعة والمساحات الحمراء الممتلئة إلى مساحات مظللة عرضية الشكل (Cross – Hatch-Pattern). لنأخذ مثال آخر. معظم شاشات المحطات الطرفية ليست مربعة، لذا يختلف طول أحد الضلعين- الأفقي أو العمودي. وهذا يعتمد على نسبة مربع أقصى (Aspect Ratio) الذي هو نسبة الطولين، سوف نعمل اختيارات مختلفة فيما يتعلق بحجم وتعيين مكان مفردات الرسومات (Graphical Items) كقائمة

الاختيارات. لهذا يتضح أن الدوال الاستعلامية تساعد المستفيد باكتشاف أية خواص متعلقة بمحطات العمل أو الحالة الداخلية للمنظومة .

4.3 برنامج بسيط (Simple Program)

لو بدأنا مع برنامج بسيط تستخدم فيه عدد قليل من دوال الإخراج فقط. فعلى وجهة التحديد، مثل هذا البرنامج لا يمكن أن يشتغل، لأن كل برنامج ينبغي أن يحتوي على عبارات تحكم وشروط رؤية محددة. نستطيع تجنب هذه القضايا الآن وذلك باستخدام منهجيتنا في تصميم البرمجيات وتحديد القيم الافتراضية (Default Settings) من قبل منظومة الرسومات.

1.4.3 نموذج الراسم القلمي (The Pen-Plotter Model):

سوف نبحث برنامج نموذجي للإخراج ذو اتصال غير مباشر بالحاسوب (Offline). غالباً ما يدعى هذا النوع من برمجة الرسومات "مناظرة الراسم القلمي " (Pen-Plotter Anology)، لأنه مطابق إلى أسلوب الرسومات التي سبقت محطات العمل الحديثة للرسومات المتفاعلة (Modern Interactive Graphics Workstation). إذن، نحصل على الرسومات بواسطة كتابة برنامج يتم تسليمه إلى معالج دفعات (Batch Processor). وبعده يتم تنفيذ البرنامج إذا كان خالياً من الأخطاء وفي وقت لاحق يتم إنتاج الرسومات على جهاز كالراسم القلمي. هكذا العملية هنا لم تتضمن أي مفاعلة (Interaction)، والأوامر المتاحة للمستفيد عادة تعتمد على أنواع من الكيانات التي يمكن إخراجها على الراسم القلمي: كالخطوط والنصوص. أيضاً الصفات المميزة المرافقة لهذه الكيانات الأولية كانت أيضاً بسيطة، فمثلاً أنواع قليلة من الخطوط، ولربما عدد قليل من اتجاهات النص (Text Directions). هنالك جانب مهم لهذه الطريقة في إنتاج مخرجات الرسومات هي حالما تنتج المخرجات، لا توجد طريقة لتغيير هذه المخرجات إلا بتغير وإعادة تشغيل البرنامج. بالإضافة إلى ذلك لا توجد آلية للتميز، على سبيل المثال، لربما وضع البرنامج خطا قليلاً إلى اليمين، لذا نحن نرغب بتحريك الخط وإعادة رسمه. إذن لتحريك هذا الخط، ينبغي علينا أن ندخل بأي طريقة الموقع الجديد للخط إلى البرنامج الأصلي، بدلاً من تغيير المخرجات المنتجة من قبل ذلك البرنامج. إن معظم تطبيقات رسم البيانات لا تزال هذه الصيغة بالرغم من توفر مكونات مادية وبرمجية أكثر فعالية لأداء مثل هذه المهمة.

إن هذا النموذج في طريقة إنتاج صور مولدة – بالحاسوب لا تختلف كثيراً فيما لو أردنا أن نرسم الصورة باليد، لنأخذ بنظر الاعتبار إجراءات الرسم التالية:

```
draw- picture (     )
{
      select – paper(  );
      orient- paper(  );
      select –pen(  );
      draw – lines(  );
      remove-paper(  );
}
```

وبلغة الإجراءات أو دوال الرسومات، كل خطوة من هذه الخطوات مطابقة إلى أحد صنوف الدوال التي شرحناها في البند السابق، لنرى ما يلي:

الخطوة الأولى: مرحلة التمهيد أو البدء (التحكم أو السيطرة).

الخطوة الثانية: مع قليل من الخيال الواسع هذه المرحلة تطابق اختيار شروط المشاهدة أو الرؤية . ينبغي علينا أن نقرر في أي اتجاه تكون الورقة إلى الأعلى. أيضاً قد نرغب استخدام جزء من الصفحة فقط، لأنه قد نرغب إضافة صورة أخرى لاحقاً على الأجزاء الأخرى من الصفحة.

الخطوة الثالثة: نقوم باختيار قلم حسب اللون والسمك المطلوب لأنها تعتبر صفات مميزة للخطوط والنصوص.

الخطوة الرابعة: رسم الخطوط والنصوص.

الخطوة الخامسة: وأخيراً تنهي الإجراء (وهو أيضا تحكم).

وهكذا، معظم تطبيقات الإخراج فقط لها انسيابية مشابهة إلى ما يلي:-

```
graphics – program  (  )
{
      initialization (   );
      set – viewing – conditions (  );
      generate- primitives (    );
      termination (    );
}
```

في الواقع تكون مرحلتي التمهيد والإنهاء في معظم البرامج متماثلة. سوف نقوم بتأجيل مناقشة دوال التحكم الضرورية حيث سنضع جميع هذه العمليات في إجرائين ندعوهما finish, init. نظراً لتكرار استخدام هذه الإجراءات، سوف نقوم بتعريفها كإجراءات مستعارة (Dummy Procedures).

نستطيع تجنب مناقشة الصفات المميزة (Attributes) وشروط الرؤية باستخدام القيم الافتراضية (Default) لها. لو ارتئينا تفحص ما في داخل مكتبة الرسومات (Graphics Library) لوجدنا هناك عدد من الجداول والمعلميات. بعض منها تتعلق بشروط التحكم، مثل أي من محطات العمل تكون مفتوحة وما هي حالة الخطأ. وقد تكون هناك معلميات أخرى تتعلق بالصفات المميزة (سعة عرض الخط الحالي، طاقم حروف طباعة النص الحالي) وشروط الرؤية (تقليم النافذة، بوابة الرؤية). بالنسبة لكثير من هذه الشروط (لا تشمل جميعها) ، تقوم معظم منظومات الرسومات بتثبيت قيم البدائل الافتراضية أثناء تمهيد المنظومة. إن أحد فوائد استخدام القيم الافتراضية هو ليس كل برنامج يحتاج أن يقوم بتثبيت كل معلمية. يكون من الطبيعي، أن يتم إعداد هذه المعلميات لتأخذ قيم افتراضية معقولة لو أردنا أن تكون هذه الوسيلة نافعة. على سبيل المثال، تكون عادة القيمة الافتراضية لصفات الخط هي متصل (Solid) ورفيع (Thin). لو أخفقنا بتثبيت المعلميات بصورة صريحة ، قد لا نجد أية مشكلة مع البرامج التمهيدية البسيطة.

2.4.3 متعدد الخطوط والنص (Polyline And Text)

لنبدأ مع كيانين أوليين هما متعدد الخطوط والنص. سوف نستخدم البديل الافتراضي لنسق النص (الوجه والطاقم – Face And Font)، وحجم الرمز واتجاه النص (يسار أو يمين) والبديل الافتراضي لنوع الخط. إن أسمى هذين الإجرائين في لغة C المستخدمين في منظومة GKS هما gtext, gpolyline ، وينبغي أن يشابها إلى ما هو موجود في منظومات أخرى.

يتم الوصول إلى دالة polyline بواسطة الإجراء التالي:

```
void gpolyline (num-pt, array-pt);
Gint num-pt;
Gpt  * array – pt;
```

إن array – pt عبارة عن مصفوفة ذات بعدين و num-pt عدد النقـــاط. لقــد تم تعريف نوع أساسي للبيانات Gpt بواسطة زوج من البيانات Gfloats

```
typedef struct
{
      Gfloat  x;
      Gfloat  y;
}     Gpt ;
```

كما جاء في الفصل السابق.

لنفترض أن النقاط في gpolyline مطابقة إلى قيم y,x المحددة في برنامج المســتفيد، كما هو مبين من خلال قطعة البرنامج التالية:

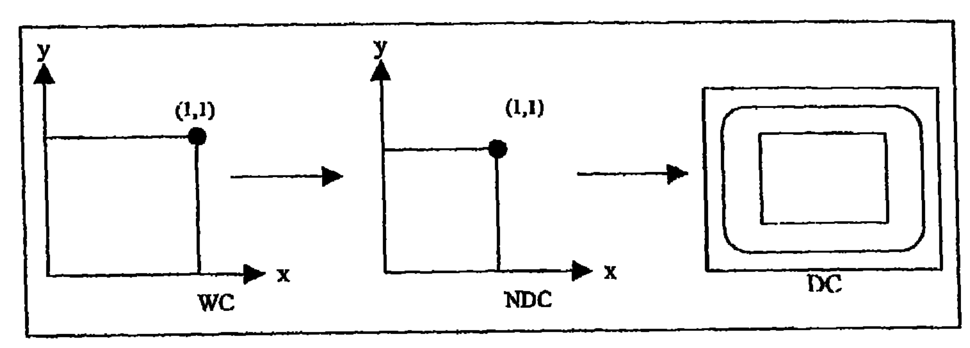
```
for (i=o; i < n, i++)
{
      points [i]. x = x [i];
      points [i].y = y[i];
}
```

من هنا، عند استدعاء gpolyline يتم تعريف قطعة خط واحدة من النقطة (x_0,y_0) إلى (x_1,y_1)، وقطعة أخرى من (x_1,y_1) إلى (x_2,y_2)، وهكذا ، وينتهي مع قطعة الخط مــن (x_{n-1}, y_{n-1}) إلى (x_n,y_n). تكون القيم الموجودة في المصفوفة المسماة points في منظومـــة إحداثيات WC. في أي حال، متعدد الخطوط المعرف أو أي جزء منه الذي سيتم عرضـــه على محطة عمل يعتمد على شروط الرؤية الفعالة حالياً عند تعريف متعدد الخطوط.

الآن الحد الأدنى لبرنامج قد يبدوا هكذا:

```
main (   )
{
      static gpt data [2] = {{1.0, 1.0}, {2.0,2.0}}
      init (  );
      gpolyline (2,data);
      finish (    );
}
```

يقوم هذا البرنامج بتعريف قطعة خط واحدة من (1.0,1.0) إلى (2.0,2.0). لكن لو أردت تشغيل هذا البرنامج وذلك باستخدام الإجراءين init, finish اللذان سنقوم بشرحهما لاحقاً في هذا الفصل سنجد مخرجاته صفحة فارغة أو ظهور شاشة فارغة. هذا ناتج بسبب شروط الرؤية الافتراضية. لأننا لم نحدد النافذة وبوابة الرؤية، لهذا منظومة GKS تبدأ مع القيم الافتراضية للنافذة وبوابة الرؤية. إن النافذة الافتراضية عبارة عن مربع وحدة (Unit Square) في منظومة إحداثيات WC، مع نقطة الأصل عند الزاوية السفلى لليسار. وتكون بوابة الرؤية أيضا مشابهة إلى مربع وحدة في منظومة إحداثيات NDC. يتم نقل هذه المنطقة من فضاء NDC إلى أكبر مربع موجود على محطة العمل الحقيقية كما مبين في الشكل 7.3.



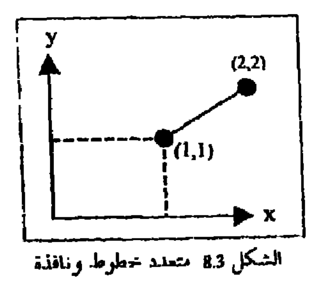
الشكل 7.3 تحويلات البديل الافتراضي

نستطيع أن نرى من الشكل 8.3 ذلك، بالرغم أن متعدد الخطوط الذي تم تعريفه تواً يعتبر متعدد خطوط مقبول بصورة صحيحة، لكنه يقع خارج النافذة ولا يمكنه توليد أي مخرجات. من الطبيعي، لو أبدلنا البيانات التمهيدية لـ data بواسطة

static Gpt data [2] = {{ 0.0,0.0},{1.0, 1.0}};

سوف نشاهد متعدد الخطوط على محطة العمل.

إنه من المعقول تماماً تركيب إجراءات أخرى باستخدام الكيان الأولي متعدد الخطوط. لنأخذ على سبيل المثال، إجراء رسم الصندوق (مربع أو مستطيل):



الشكل 8.3 متعدد خطوط ونافذة

```
box (xmin, xmax, ymin, ymax)
Gfloat xmin, ymin, xmax, ymax;
{
      Gpt points [5];
      points [0] . x = xmin;
      points [0].y = ymin;
      points [1].x = xmax;
      points [1] .y= ymin;
      points [2].x=xmax;
      points [2].y=ymax;
      points [3].x=xmin;
      points [3].y=ymax;
      points [4].x =xmin;
      points [4].y =ymin;
      gpolyline (5, points );
}
```

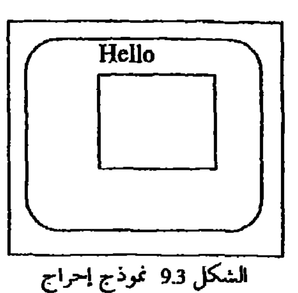
من أجل إعطاء برنامجنا هذا أهمية أكبر، نستطيع إضافة كيان أولي للنص من خلال الإجراء التالي:

```
gtext (start, string)
Gpt start;
Gchar *string
```

هنا يمثل start موقع بداية النص المسمى string. تكون احداثيات النقطة start في فضاء WC وكذلك شروط البديل الافتراضي لرموز النص هو أن يكون ارتفاع الرمز 0.05 وحدة في فضاء WC، وتبدأ سلسلة الرموز من اليسار أي اليمين من نقطة البداية start عند الزاوية السفلى لليسار لسلسلة الرموز (string).

الآن ، برنامج بسيط قد يبدأ هكذا:

```
main (   )
{
      statatic Gpt start = {0.25 , 0.9};
      init (   );
      box (0.25, 0.75 , 0.25 , 0.75);
      gtext (&start, "Hello");
      finish (   );
}
```

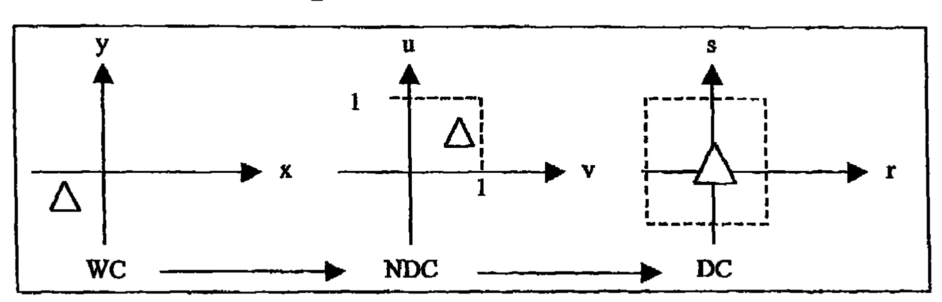


الشكل 9.3 نموذج إخراج

الذي يقوم بإنتاج مخرجاته كما مبين في الشكل 9.3 كل هذا قد يبدوا بسيطاً. هذا صحيح! الآن مع كيانين أولين فقط، يمكننا توليد مخططات بيانية (Graphs) ورسوم بيانات (Plots) ومخططات إنسيابية (Flowcharts) ورسوم الدوائر الكهربائية. قد ترغب الأخذ بنظر الاعتبار كيفية التحكم في مركزه النص (Center Text) على الشاشة والقيام ببعض المعالجات الأخرى لأجل إظهار المخرجات بصورة جيدة. ما سنجده في البنود القليلة القادمة هو توفر وسائل عديدة في منظومة الرسومات تجعل مهمة إنتاج المخرجات المعقدة أو المركبة سهلة الإنجاز.

5.3 الرؤية أو المشاهدة *(Viewing)*

نستطيع تعريف الرؤية بدلالة التحويلات بين منظومات الإحداثيات. لقد رأينا سابقاً أن المستفيد يتعامل مع منظومة إحداثيات كونية (World-Coordinate-WC). داخلياً معظم المنظومات تستخدم إحداثيات تعيير الجهاز (Normalized Device Coordinate- NDC)، بينما كل محطة عمل حقيقية تستخدم منظومة إحداثيات الجهاز (Device Coordinate- DC) الخاص بها. إذن، كما رأينا أن هناك تحويلان ضروريان، الأول من WC إلى NDC (التحويل المعياري – The Normalization Transformation) والثاني من NDC إلى DC (تحويل محطة العمل – The Workstation Transformation) . إن كل من المنظومات الإحداثية ثنائية الأبعاد لها نقطة أصل خاصة بها كما هو موضح في الشكل 10.3.



الشكل 10.3 إحداثيات كونية WC وإحداثيات تعيير الجهاز NDC

1.5.3 التحويل القياسي أو المعياري (The Normalization Transformation)

نحن لم نعطي معلومات كافية لتعيين تحويل وحيد أو النقل بين أي زوج من منظومات الإحداثيات هذه. نحن نعلم أن جزء معين من فضاء WC سيتم نقله إلى حيز معين في فضاء NDC، وكذلك سيتم نقل حيز معين من فضاء NDC إلى حيز على محطة العمل (DC).

نحن نفترض هنا وجود محطة عمل واحدة فقط لتجنب الإرباك. في الفصل السابق، اعتبرنا منظومة إحداثيات NDC تمثل شاشة افتراضية أو ظاهرية (Virtual Screen). يكون سطح العرض لهذا الجهاز عبارة عن مربع إحداثيات زاويته السفلى- لليسار في منظومة NDC هي (0.0 , 0.0) وإحداثيات زاويته العليا – لليمين هي (1.0, 1.0) . نستطيع استخدام أي جزء من هذه المساحة. تعتبر المنطقة المختارة بمثابته بوابة الرؤية. في منظومة إحداثيات WC، نقوم باختيار مساحة (النافذة – Window) ليتم نقلها إلى بوابة الرؤية هذه. نقوم بتحديد النافذة وبوابة الرؤية معاً وذلك بإعطاء إحداثيات الزاويتين للمستطيلين المعرفين وهما الزاوية السفلى لليسار والزاوية العليا لليمين اللتان تعرفان النافذة والبوابة. إذن النافذة عبارة عن مستطيلة الشكل معرفة في منظومة إحداثيات WC يتم نقلها إلى مستطيل معرف في منظومة إحداثيات NDC.

في كثير من التطبيقات يكون مناسباً أن تكون لدينا القدرة من استخدام أكثر من منظومة إحداثيات WC واحدة وأيضاً أكثر من بوابة رؤية واحدة. على سبيل المثال في منظومة متفاعلة مسيرة بقائمة اختيارات (Menu-Driven Interactive System)، يتم إظهار قوائم الاختبار ومخرجات الرسومات على بوابات رؤيه مختلفة. وربما يتم تعريف كل واحدة منها مستخدمين منظومة إحداثيات WC التي تكون أكثر ملائمة للمشكلة. إذن، لكل نافذة وبوابة رؤية محددة سيعرف لها تحويل معياري وحيد. وبدلاً من إعادة تعريف هذه التحويلات بصورة مستمرة ضمن البرنامج، نفضل السماح بتواجد تحويلات معيارية متعددة في وقت واحد. ويتم تحقيق هذه المهمة وذلك باستخدام معلمية (Parameter) إضافية، تدعى مؤشر أو دليل التحويل المعياري (Normalization Transformation Index) في مواصفات النافذة وبوابة الرؤية. إذن لتعريف نافذة وبوابة رؤية، يكون لدينا الإجرائين التاليين:

```
void gset-win (trans-num, window)
void gset-vp (trans-num, viewport)

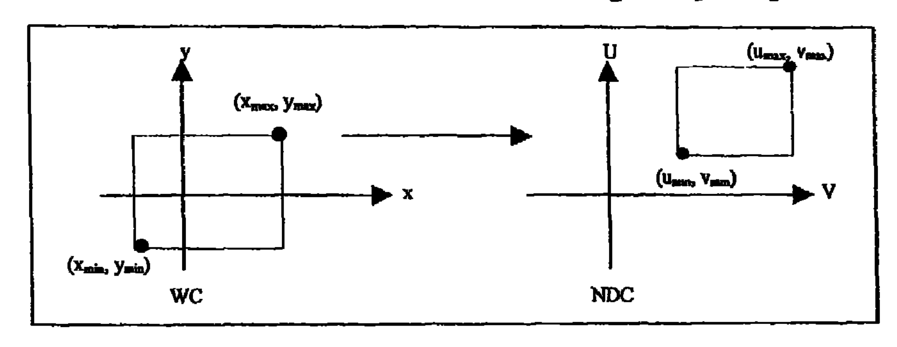
Gint trans-num;
Glim  *window, *viewport;
```

هنا اعتبرنا trans-num هو دليل التحويل المعياري. يتم تعريف النافذة وبوابة الرؤية بواسطة زوجين من الإحداثيات هما:

(x_{min}, y_{min}) , (x_{max}, y_{max}) and

(u_{min}, v_{min}) , (u_{max}, v_{max}),

كما هو مبين في الشكل 11.3.



الشكل 11.3 تعريف تحويل المعياري

ومن خلال اثنين من التراكيب في الصيغة التالية:

```
typedaf struct

        Gfloat xmin;
        Gfloat xmax;
        Gfloat ymin;
        Gfloat ymax;
   }Glim;
```

عادة، البديل الافتراضي لشروط الرؤية الذي سبق استخدامه في البند السابق يطابق التمويل المعياري '0' المعرف مسبقاً (وغير قابل للتغيير) كمربع وحدة في كلا الفضائين WC و NDC . يكون لهما نفس التأثير كما لو أننا كتبنا العبارات التالية:

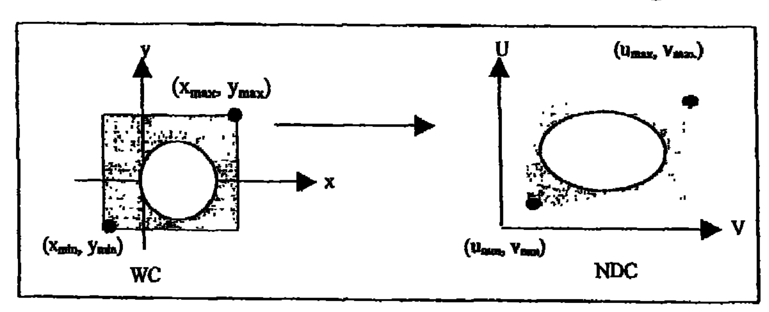
```
static glim window0 = {0.0 , 1.0, 0.0, 1.0};
static glim viewport0 = {0.0 , 1.0, 0.0, 1.0};
gset – win (0,&window0);
gset – vp (0,&viewport0 );
```

يتم اختيار النافذة وبوابة الرؤية المطلوبة من خلال الدالة "اختار تحويل معياري" (select normalization transformation):

```
void gsel- norm-tran (trans-num)
Gint trans-num;
```

يبقى التحويل ساري المفعول حتى يتم تغييره ويطبق التحويل المعياري الحاضر على جميع الكيانات الأولية. هكذا، نستطيع تعريف مجموعة من التحويلات المعيارية مستخدمين gset-win و gset-vp في بداية البرنامج أو ضمن إجراء، ويمكن التنقل بينهما من خلال gsel-norm-tran.

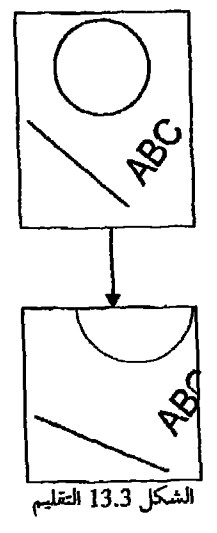
لا يوجد هنالك أي تقييد على حجم النافذة الكونية (World Window). ولكن بوابة الرؤية تكون مقيدة لأنها تقع ضمن حدود مربع وحدة معرفاً فضاء NDC. يتم نقل الكيانات الأولية المعرفة ضمن النافذة إلى منطقة سارية المفعول من فضاء NDC. ليس من الضروري أن تحتفظ هذه التحديدات بحجوم الصور، لأن نسبة المربع الأقصى (Aspect Ratio) للنافذة الكونية هي $(x_{max}-x_{min})$ / $(y_{max}-y_{min})$ قد لا يكون مساوياً إلى ما يقابله لبوابة الرؤية $(v_{max}-v_{min})/(u_{max}-u_{min})$. ما لم تكون هاتين النسبتين متساوية، ينبغي سحب الجسم المنظور بمقادير مختلفة على امتداد محوريهما x و y لكي يتطابق مع بوابة رؤيته المخصصة. مع أن الخطوط تبقى خطوط ولكن المربعات لربما تتحول إلى مستطيلات والدوائر إلى قطوع ناقصة أو أهليجية (Ellipse) كما هو مبين في الشكل 12.3.



الشكل 12.3 إطالة بوابة الرؤية

2.5.3 التقليم (Clipping).

الكيانات الأولية التي تقع داخل النافذة سوف يتم نقلها إلى بوابة الرؤية، بينما تلك التي تكون جزئياً أو كلياً خارج النافذة، يجب أن تُقلم (Clipped). تقوم عملية التقليم بتقرير أي جزء، (إن كان هناك جزءٌ) من الكيانات الأولية تقع داخل النافذة. أثناء عملية التقليم، الكيانات الأولية التي تقع خارج النافذة تهمل وأما الكيانات الأولية التي تكون جزئياً داخل النافذة يتم تقطيعها إلى أجزاء، لكي تعرض الأجزاء داخل النافذة فقط كما مبين في الشكل 13.3.



الشكل 13.3 التقليم

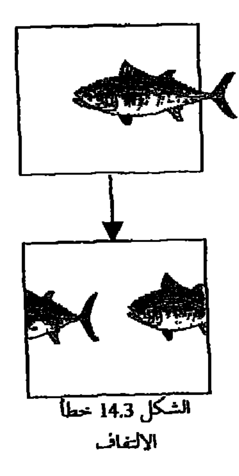
سنقوم بشرح خوارزميات التقليم عندما نأخذ بنظر الاعتبار قضايا التنفيذ في الفصل السادس.

أما في الوقت الحاضر، علينا إدراك أن التقليم هو عملية داخلية ضرورية لأي حزمة برامج رسومات.

غالباً ما يخطأ المبتدئيون بافتراض أن المكونات المادية سوف تعتني بعملية التقليم. عادة هذا الافتراض يكون مبيناً على فكرة أن القيم التي تقع خارج النافذة يتم نقلها إلى قيم كبيرة جداً للمكونات المادية، إذن هذه القيم يتم تحديدها ذاتيا بواسطة قيم مساوية للحد الأعلى والأدنى المسموح من قبل المكونات المادية.

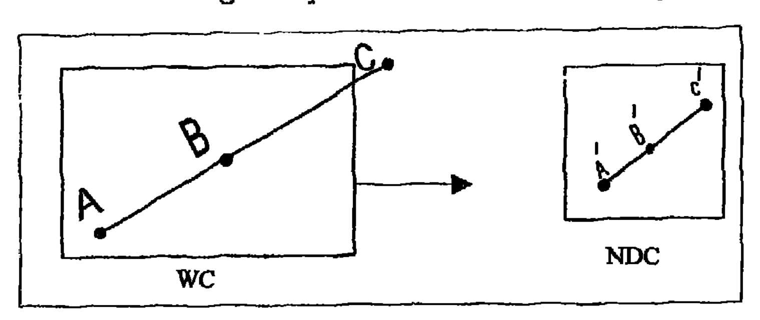
توجد هنالك مشكلتان في هذا الإدعاء وهما:

1- قد لا تكون المكونات المادية قادرة على التعامل مع القيم التي تكون كبيرة جداً أو صغيرة جداً. مع المكونات المادية القديمة، عندما تكون القيم كبيرة جداً تؤدي إلى ظاهرة الالتفاف (Wrapped Around) عندما يتسبب بفيض مسجلات المكونات المادية (Hardware Registers)، ولذلك قد يؤدي بالكيانات الأولية أن تظهر في مواقع غريبة على الشاشة كما هو مبين في الشكل 14.3.


الشكل 14.3 خطأ الالتفاف

2- إن هذا الإدعاء يتجاهل عملية نقل النافذة -إلى- بوابة

الرؤية . على سبيل المثال، لنأخذ الخط ABC المبين في الشكل 15.3.



الشكل 15.3 التقليم بالنسبة للنافذة

قطعة الخط AB تقع داخل النافذة وقد تم نقلها إلى بوابة الرؤية المناظرة لهـــا. وأمـــا قطعة الخط BC الواقعة خارج النافذة ينبغي تقليمها. مع ذلك، نظراً لكون بوابة الرؤيـــة أصغر من مربع الوحدة (Unit Square) التي تعـــرف شاشـــتنا الافتراضيـــة (Virtual Screen). لذا يتم نقل قطعة الخط BC إلى منطقة سارية المفعول من فضاء NDC، بالرغم من أن هذه المنطقة تقع خارج بوابة الرؤية المطلوبة. هذه الحالة هي إحدى الحالات الـــتي عادةً لا يمكن تركها للمكونات المادية. عادة تقوم المكونات المادية بتدقيق فيما إذا كـــان الكيان الأولي يقع في مدى الحدود الخاص به، وليس كونه يقع ضمن حدود بوابة الرؤيـــة المعرفة من قبل المستفيد.

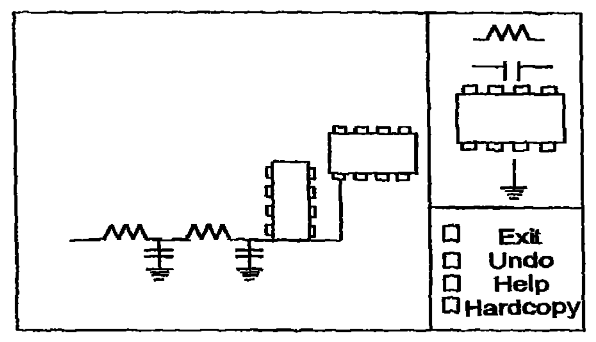
هناك طريقة أخرى يمكننا النظر إلى هذه الحالة وذلك أن ندرك أن كثيراً من أجـــهزة المكونات المادية كالمحطات الطرفية، لا تمتلك أي فكرة عن ماذا تكون بوابة الرؤيـــة. إذن ينبغي على البرمجيات ضمان ذلك، حيث يتم تقليم الكيانات الأولية للإخراج بالنســـبة إلى بوابة الرؤية المعنية قبل إرسالها إلى المكونات المادية للعرض.

هنالك حالات نرغب فيها بإيقاف عملية التقليم هذه لأنها تستهلك وقتاً. لنضرب مثـــلاً، قد نستخدم مكونات مادية تدرك النوافذ وبوابات الرؤية أو قد تعلم مسبقاً أن كياناتنا الأوليـــة تقع داخل النافذة تماماً ولذلك لا يكون التقليم ضرورياً. في هذه الحالات وجود إجراء يحجـــب عملية التقليم يكون مفيداً، لذلك تم إضافته إلى كثير من منظومات الرسومات .

3.5.3 تحويل محطة العمل (The Workstation Transformation)

الآن نستطيع بحث التحويل الثاني وهو تحويل محطة العمل. ينبغي نقل الأشياء المنظورة (Objects) في فضاء NDC إلى فضاء إحداثيات الجهاز (DC) لتكوين عروض على أجهزتنا الحقيقية. لو قمنا بتحديد التحويل لكوني (World Coordinate) فقط، سيقوم البديل الافتراضي لتحويل محطة العمل بنقل جميع نقاط فضاء NDC إلى أكبر مربع ينطبق مع عارضة محطة العمل. يكون هذا الاختيار معقول لكونه سيحافظ على نسبة المربع الأقصى (Aspect Ratio)، وحيث لا يكون هناك أي تشويه للأشياء المنظورة بين فضاء NDC ومحطة العمل (فضاء DC). في كثير م البرامج يكون البديل الافتراضي هذا مقنعاً. مع ذلك، وجود مرونة في تغيير هذا الاختيار يمكن أن يكون نافعاً كما مبين في المثال التالي.

لنأخذ بنظر الاعتبار مرة ثانية برنامج مخطط الدائرة الكهربائية الذي تم فيه تنظيم العارضة كما مبين في الشكل 16.3.



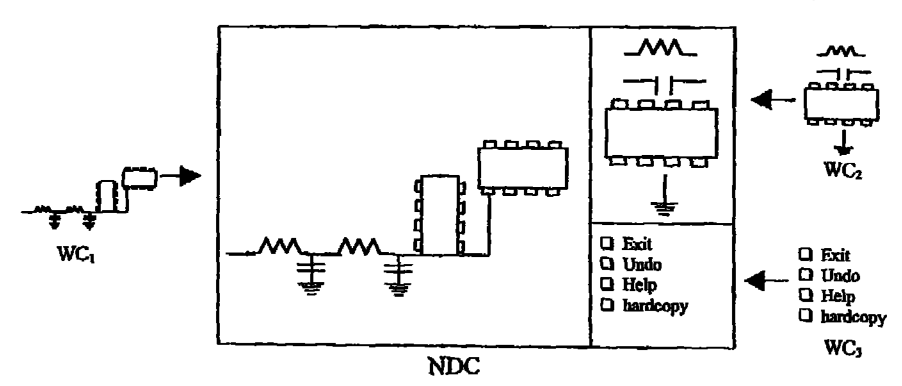
الشكل 16.3 عارضة مخطط الدائرة الكهربائية

هنا نقوم بتحديد مساحة واحدة لقائمة اختيار المركبات التي يمكن استخدامها، ومساحة أخرى لقائمة اختيار أوامر السيطرة ومساحة كبيرة للمخطط نفسه. مع أننا لا يمكن لحد الآن مناقشة الكيفية التي يستطيع المستفيد أن يتفاعل مع هذه الصورة، لكن عادة تكون أول خطوة في كتابة مثل هذا البرنامج هو القيام بتصميم مخطط العرض الذي سيشاهده المستفيد على شاشة المحطة الطرفية. لقد رأينا سابقاً، بالنسبة إلى منظومة الرسومات، تعتبر هذه الشاشة محطة عمل منطقية واحدة. عند مرحلة معينة أثناء عملية

التصميم، قد يرغب المستفيد أن يحصل على نسخة مطبوعة للدائرة التي تم تصميمها. من المعتاد، يتم رسم الدائرة الكهربائية المصممة على راسم البيانات (Plotter) فقط ولا حاجة لرسم قوائم الاختيار. بالنسبة إلى منظومة الرسومات، يعتبر راسم البيانات محطة عمل منطقية ثانية.

إذن ، ينبغي ظهور عرضان مختلفان على محطتي العمل، وبقدر ما يتعلق الأمر بالبرنامج، يجب ظهورهما في آن واحد. بالإضافة، قد يستخدم هذا البرنامج ثلاثة تحويلات معيارية مختلفة. حيث يتم نقل قائمتي اختيار من WC إلى بوابتين مختلفتين للرؤية في فضاء NDC، وتستخدم مساحة مخطط الدائرة الكهربائية بوابة رؤية ثالثة.

جميع هذه التحويلات موضحة في الشكل 17.3.



الشكل 17.3 تحويلات تصميم الدائرة الكهربائية

نحن نرغب نقل كل ما نراه في فضاء NDC إلى محطة العمل الطرفية (Terminal Workstation)، ولكن فقط جزء منه يتم نقله إلى محطة العمل لراسم البيانات (Plotter Workstation).

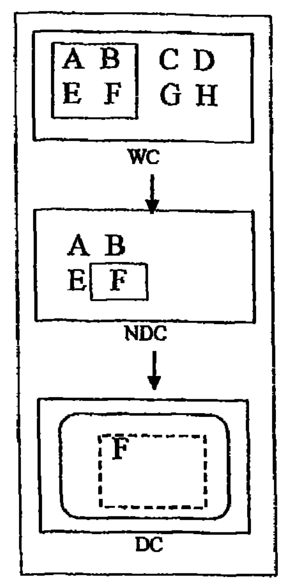
الإجراءان اللذان يعرفان هذا النقل هما:

1- إعداد نافذة محطة العمل void gset – ws-win (wk-id, ws-window)

2- إعداد بوابة رؤية محطة العمل void gset – ws- vp (wk-id, ws-viewport)

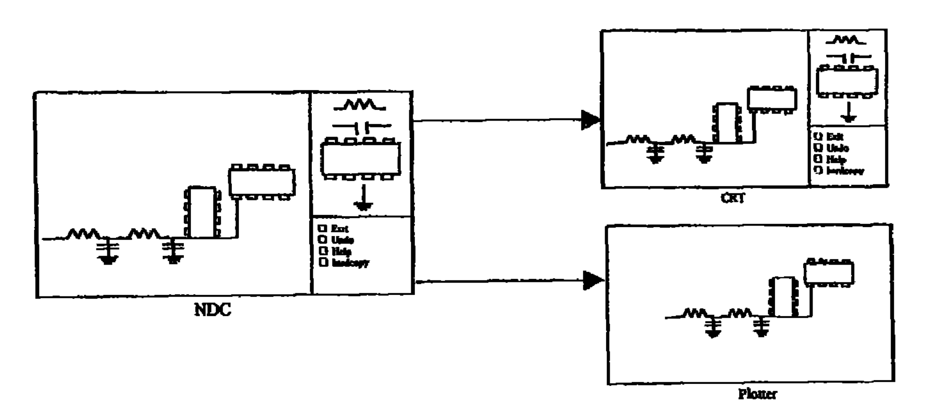
```
Gint wk-id;
Glim *ws – window;
Glim *ws-viewport;
```

يعرف gset – ws-win نافذة محطة عمل الـذي يحمـل الاسم المنطقي لمحطة العمل wk-id. يكون رمز تعريف محطـة العمل هذا قد تم تثبيته في إجراء التمهيد init. هذه النافذة تحدد المنطقة من فضاء NDC التي ستنقل إلى محطة العمـل المشـار إليها. أيضاً gset-ws-vp تعرف بوابة رؤية على محطة– وهـي عبارة عن مساحة إليها يتم نقل محتوى نافذة محطة العمل. نظراً لكون نافذة محطة العمل هي مساحة في فضاء NDC لذلك يتم تحديد معلمياته في فضاء NDC. أما بوابة رؤية محطة العمـل فهي تكون على محطة العمل وبالتالي يتم تحديد إحداثياتها في فضاء إحداثيات DC. بالنسبة إلى التحويل المعياري وتحويـل محطة العمل، يتم نقل نافذة في أحد الفضـائيين إلى بوابـة في الفضاء الآخر، كما هو مبين في الشكل 18.3.



الشكل 18.3 الانتقال من فضاء أحداثيات الكروية إلى فضاء أحداثيات محطة العمل

لنعود إلى مثال برنامج مخطط الدوائر الكهربائية، نستطيع تعريف تحويلين محطة عمل كما مبين في الشكل 19.3.



الشكل 19.3 انتقالات محطة العمل لبرنامج مخطط الدائرة الكهربائية

ولكن هناك مشكلة صغيرة. لأول مرة ، نواجه مشكلة تحديد معلميات في منظومـة DC. عند هذه النقطة، يمكننا التخلي عن مفهوم الاسـتقلالية عـن الجـهاز (Device

(Independence والالتجاء إلى الاستعانة بدليل يدوي (Manual) لإيجاد أبعاد العارضة في منظومة DC. لكن البحث عن هذه المعلومة ليس ضرورياً، حيث يمكننا اســـتخدام دوال استعلامية (inquiry) للحصول على هذه المعلومة. نظراً لكون هذه المعلومة متوفرة مـــن خلال منظومة الرسومات، إذن لا يزال باستطاعتنا أن نكتب برامجنا بطريقة مستقلة عـــن الجهاز.

6.3 التحكم أو السيطرة (Control)

حان الوقت لدراسة محتويات الإجراءين finish, init في برنامجنا البسيط (البند 4.3). في معظم المنظومات، تشمل عملية التمهيد والإنهاء عدد من الخطوات . بالرغم أن هـــذه الخطوات قد تختلف في خصوصياتها ويتم إخفاء التفاصيل عن المستفيد، مع ذلك تكـــون الحاجة إلى التمهيد والإنهاء ضرورية لأداء العملية بصورة صحيحة. ســـنقوم باســـتعراض إجراءات منظومة GKS، لكن يبقى علينا أن نتذكر، حالما نتعرف على هذه الإجـــراءات يمكننا وضع هذه الخطوات في إجراءات مثل finish, init واللذان يتكرر اســـتخدامها. في منظومات أخرى، يمكننا تعويض كل خطوة على انفراد بما يقابلها في المنظومة الجديدة.

1.6.3 التمهيد (Initialization)

تقوم عملية التمهيد بعدد من المهمات لتهيئة المنظومة لتنفيذ برنامج المستفيد. عـــادة تتضمن هذه المهمات ما يلي:

- إعداد جداول الحالة للمنظومة (Setting Up State Table For The System)
- تخصيص ذاكرة (Allocating Memory)
- تخصيص أجهزة حقيقية (Allocating Physical Devices)
- الإعداد لبدأ في تناول الأخطاء (Initializing Error Handling)
- الإعداد لبدأ الأجهزة الحقيقية (Initializing Physical Devices)
- ربط الأجهزة المنطقية بالأجهزة الحقيقيـــة (Linking Logical Devices To Physical Devices)
- إعداد قيم البدائل الافتراضية (Setting Default Values)

كما تم بحثه سابقاً، تقوم منظومة الرسومات الحديثة بعدة مهمات كانت سابقاً تعتبر من مسؤوليات مبرمج التطبيقات . من أجل القيام بهذه المهمات ينبغي من منظومة الرسومات أن يعد للبدء ويتابع كثير من الجداول والمعلميات (Tables And Parameters). بعض القيم تمثل الحالة الداخلية للمنظومة. ومن الأمور الأخرى، يتم تحديد قيما إذا كانت بعض العمليات التي يقوم بها المستفيد سارية المفعول أم لا. على سبيل المثال قد لا يمكن إرسال كيانات أولية للإخراج إلى محطات عمل إذا لم يتم تعريف محطات العمل مسبقاً، وهذه حقيقة تحددها حالة المنظومة (State Of The System). يتم تخصيص ذاكرة لتخزين كيانات مثل تحويلات وجداول معرفة من قبل المستفيد. وكجزء من هذه العملية أيضاً يتم تثبيت قيم البدائل الافتراضية للصفات المميزة (Attributes) ومفردات مثل التحويل المعياري '0'.

بالرغم أن تدقيق الخطأ هو جزء مهم من المنظومة، ينبغي على المستفيد أن يتخذ قرار ما سيفعله عند ظهور الأخطاء. سنقوم باستخدام آلية بسيطة لتسجيل الأخطاء في ملف معد للبدء بدقة. إن إعداد أي بديل لآلية تتناول الأخطاء ينجز عادة كجزء من مرحلة التمهيد.

أيضاً كجزء من عملية التمهيد، ينبغي نقل الأجهزة المنطقية إلى الأجهزة الحقيقية بطريقة تكون مشابهة كثيراً إلى تلك الطريقة التي يتم فيها تعيين الملف المنطقي (Logical File) إلى الملف الحقيقي (Physical File) في برنامج بلغة C أثناء فتح الملف. تتطلب عملية تمهيد الأجهزة الحقيقية أن تكون تلك الأجهزة متوفرة ومعدة بدقة. على سبيل المثال، الراسم القلمي يجب أن يحتوي على ورق، في حين أنبوبة الأشعة الكاثودية CRT ينبغي أن تكون الشاشة ممسوحة.

في منظومة GKS تحتاج هذه العمليات على الأقل إلى أربعة خطوات، كما مبين في هذا الإجراء البسيط:

```
init (   )
{
    int fd ;
    fd = create ("errors", "w");
    gopen – gks (fd, BUFFER);
    gopen-ws (WS-ID, CON-ID,WS-TYPE);
    gactivate – (WS- ID);
}
```

2.6.3 ملف الأخطاء (The Error File)

يكون أول فعل مهم يتخذه إجراء التمهيد هو خلق ملف أخطاء يقبل الكتابة فيه يدعى "errors" وهو ملف قياسي في لغة C مع واصف ملف fd (file descriptor). بما أ، كل دالة رسومات قد تنتج رسالة خطأ، لذا ينبغي تخصيص مكان لتخزين هذه الأخطاء قبل تنشيط أي دالة رسومات. لكل دالة لها رقم وحيد ولكل نوع من الأخطاء له رقم وحيد لأجل تشخيص الخطأ والدالة. بالرغم من أننا نستطيع تصميم مناول الأخطاء(Error Handler) حسب متطلبات الزبون (من الآن سنقبل البديل الافتراضي) والذي يتضمن ببساطة كتابة رقم الدالة الذي حدث فيه الخطأ مع رقم الخطأ في ملف الأخطاء. وبعدئذ نستطيع تفحص محتوى هذا الملف بعد تنفيذ برنامجنا. من الطبيعي، باستطاعتنا استخدام جهاز إخراج قياسي أو جهاز أخطاء بدلاً من الملف لتتبع الأخطاء.

3.6.3 فتح المنظومة (Opening the System)

أما الفعل الثاني الذي يتم اتخاذه هو فتح منظومة الرسومات. هنا يحتوي الإجراء على معلمتين:

1- واصف ملف الأخطاء fd الذي تم فتحه حالياً.

2- BUFFER يمثل سعة المخزن الانتقالي المطلوبة. سنفترض أن هذه القيمة يتم تحديدها من قيم البدائل الافتراضية، أما بواسطة معرف الملفات (s) define file أو بواسطة المنظومة. بصورة عامة، تكون هناك كمية كافية متوفرة من الذاكرة (أو تستطيع المنظومة تخصيصها) لتناول البرامج البسيطة.

أما بالنسبة للبرامج الكبيرة، مثل تلك التي تقوم بتوليد عدد من متعددات الخطوط الكبيرة أو تحتوي على عدة أجزاء، قد تحتاج إلى طلب مزيد من الذاكرة وذلك عن طريق تعديل المعلمية BUFFER.

4.6.3 فتح وتنشيط محطات العمل (Opening And Activating Workstations)

نحن الآن مستعدون لفتح محطة عمل واحدة أو أكثر. عند تفعيل (Invocation) الدالة gopen-ws يتم ربط محطة العمل المنطقية التي سنستخدمها في برنامجنا مع محطة العمل الحقيقية التي ستظهر عليها المخرجات. بعد هذا الاستدعاء، نحتاج فقط استخدام

أسماء محطات عمل منطقية في برنامجنا. تتكون المعلمية الأولى (ws-id) من رقــــم (نــوع Gint) نختاره لكي يخصص كمعرف منطقي (Logical Identifier) لمحطة العمل والــــذي سنقوم باستخدامه في النداءات التالية عند الإشارة إلى محطة العمل هذه. وأما المعلميتــــين التاليتين تقوم ببناء الترابط مع محطة العمل الحقيقية. لسوء الحظ، طريقة تكوين هذا الترابط تختلف من تنفيذ إلى تنفيذ. إن إحدى الطرق لتعريف هذا الترابط هو جعــــل ws- type يشير إلى محطة عمل حقيقية معينة وجعل con-id يختار وحــــدة إدارة الجــــهاز (Device Driver). في تطبيقات أخرى قد تستخدم هذه المعلميات للإشارة إلى مداخــل (Entries) في ملف يحتوي على معلومات ضرورية حول محطات العمل الحقيقيــــة ووحــــدات إدارة الجهاز. نظراً لكون هذه العبارة البرمجية الواحدة تحتوي على معلومة الربط بــين محطــات العمل الحقيقية والمنطقية، عادة تكون العبارة الوحيدة التي قد تحتاج إلى تغيــــير في حالــة تشغيل تطبيقاتنا على منظومة أخرى. على سبيل المثال ، لو استخدمنا أسماء رمزيــــة الــتي يمكن تغييرها بسهولة في ملف (include)، آنذاك تفعيل نموذجي للدالة قد يبدوا مشابه إلى ما يلي:

```
gopen – ws (CRT, CONN- ID – 1, WK – TYPE- 4105);
```

وفي حالة تنفيذ معين قد نستخدم ما يلي:

```
#define  CRT 1
#define CONN – ID-1 "/dev/tty05"
#define WK-TYPE – 4105 52
```

حيث أن الرمز التذكيري (Mnemonic) قد يشير إلى محطة طرفية للرسومات مــــن نوع (Tektronix Model 4105) وفي هذا التنفيذ تكون محطة عمل نوع 52. وأما معرف التوصيل /dev/tty05 سيختار وحدة إدارة جهاز صحيحة لهذه الطرفيــــة وقــد اختــار المستفيد الرقم الصحيح 1 كمعرف منطقي لمحطة العمل هذه.

وأخيراً نستطيع تنشيط محطة العمل من خلال الدالة gactivate-ws.

يعتبر فتح وتنشيط محطة العمل فعاليتين مستقلتين عن بعضهما البعض، لذا لا ترســل الكيانات الأولية إلى محطة مفتوحة ولكنها غير فعالة، إن فتح محطة العمل لا تسبب فقــط فعاليات داخلية، كتثبيت قيم في جداول الحالة ولكن هنالك أيضــاً بعــض الفعاليــات

الخارجية، كمسح شاشة محطة العمل. لهذا لو أردنا اعتراض كيانات أولية من الذهـاب إلى محطة ا لعمل وكان المطلوب هو تنشيط محطة العمل لاحقاً، يفترض أن نبقيها مفتوحة وإلا قد تكون مجازفة في فقدان ما موجود حالياً على العارضة.

إن فتح محطات عمل إضافية يمكن شمولها في الدالة init. على سبيل المثال، نسـتطيع فتح وتنشيط راسم رقمي بإضافة العبارتين التاليتين:

```
gopen – ws (PLOTTER, CONN-ID-2,WS-TYPE-PLOTTER);
gactivate-ws (PLOTTER);
```

مرة ثانية ، هذه الصفوف من الرموز (strings) يفترض أن تكون معرفة في ملفـات (include) أو في برنامجنا.

5.6.3 الإنهاء (Termination)

يكون الإنهاء أبسط بكثير من التمهيد. علينا إبطال كل فعاليـة تم إتخاذهـا أثنـاء التمهيد. هنا لنرى الإجراء finish:

```
finish (  )
{
    gdactivatews (ws-id);
    gclosews (ws-id);
    gclosegks (  );
}
```

أولاً نقوم بإخماد أو إبطال فعالية (Deactivate) محطة العمل ومن ثم نغلقـه. نقـوم بنفس الإجراء مع أي محطة عمل فعالة أخرى. حالما يتم إغلاق جميع محطـات العمـل، نستطيع إغلاق المنظومة. إن هذه الفعالية تسبب حدوث عدد من الأمور داخليـاً. مـاذا سنرى قد يكون مسح لسطح شاشة الجهاز الحقيقي أو قذف ورقة من الراسم القلمي. من الممكن غلق ملف الأخطاء كأي ملف اعتيادي بلغة C أو يغلق ذاتياً عند انتهاء البرنـامج. من الممكن تدقيق هذا الملف من أي أخطاء بعد تنفيذ البرنامج.

7.3 الصفات المميزة لمتعدد الخطوط والنص (*Polyline And Text Attributes*)

الآن نعود إلى كياناتنا الأولية وإلى مشكلة كيفية عرضها. لحد الآن ، كياناتنا الأوليـة الوحيدة كانت متعدد الخطوط والنص فقط، والتي تم الوصول إليها من خلال اسـتدعاء

الإجرائين gploylinee و gtex. عند التعرف عليهما، لم يتضمن الإجرائين أية آلية تسمح لنا بتغيير لون الخطوط وإتجاه النص أو أية خواص أخرى لهذه الكيانات . هنا سوف نتعرف على معظم الصفات المميزة لهذين الكيانين الأوليين. في البندين القادمين، سوف نتعرف أخيراً على كيانات أولية أخرى في منظومة GKS وسنكتب مثال لبرنامج طويل نستخدم فيه بعض هذه الصفات المميزة.

تقوم الصفات المميزة بتحديد طريقة عرض الكيان الأولي . حيث تحدد الصفات المميزة اللون والحجم والاتجاه. لبعض الكيانات الأولية، كمتعدد الخطوط، لها عدد قليل من الصفات المميزة، وأما غيرها كالنصوص تحتاج إلى عدد كبير من الصفات لكي تسمح لنا إنتاج مخرجات عالية النوعية. سوف نتعرف على جميع الصفات المميزة في هذا البند، ولكن سوف لا نبحث كل صفة بالتفصيل. هدفنا هو أن نجعلك يقضاً بما هو ممكن. ستجد معلومات أكثر تفصيلا حول الصفات المميزة في الملحقين أ و ب.

1.7.3 صفات هندسية وغير هندسية (Geometric And Nongeometric Attributes):

نستطيع إعداد الصفات المميزة بعدة طرق . لو شاهدنا خط أخضر على المعارضة، لربما نسأل أنفسنا لماذا يكون الخط أخضر، هل أن الاخضرار هي صفة مميزة للخط والمألوف أن يكون لون الحشيش أخضر بدلاً من اللون الأزرق. أو قد يكون الخط أخضر ببساطة وذلك بسبب قرار المبرمج أن يعرض الخط باللون الأخضر في حين يمكنه بسهولة أن يعرضه باللون الأحمر أو الأزرق. هاتين الفكرتين يعكسان بالأساس أوجه نظر مختلفة في كيفية مرافقة الصفات المميزة مع الكيانات الأولية. حيث تجد في وجهة النظر الأولى، الصفة تكون مقيدة بالكيان الأولي ولا يمكن تغيرها أبداً. ومن وجهة النظر أخرى، نستطيع تبديل الصفات أثناء تنفيذ البرنامج، عادة عند مستوى محطة العمل. أي من وجهتي النظر قد تكون صحيحة ضمن سياق المشكلة المعينة. ومن حسن الحظ المنظومات كمنظومة GKS تعطينا مقدار كبير من المرونة في كيفية تناول الصفات المميزة.

الصفات المميزة لكيان أولي (Primitive Attributes) تلك التي تكون ملازمة للكيانات الأولية للإخراج بدلاً من أن تكون ملازمة إلى مجموعات من الكيانات الأولية – تقسم إلى صنفين هندسية وغير هندسية. الصفات الهندسية لها حجم وتثبت بصورة محددة في منظومة

إحداثيات WC. على سبيل المثال، ارتفاع الرمز هي صفة هندسية. الصفات الهندسية تكون ملازمة للكيانات الأولية ولذلك تكون مستقلة عن محطة العمل المستخدمة لعرضها.

كل الصفات التي لا تكون هندسية تدعى صفات غير هندسية. على سبيل المثال، نوع الخط المستخدم في متعدد الخطوط (متصل أو متقطع أو منقط) صفة غير هندسية. يتم إعداد الصفات غير الهندسية بواسطة مؤشرات في الجداول. بعض المداخل في هذه الجداول تكون معرفة مسبقاً. إذن جميع الصفات بالإمكان تثبيتها من قبل المبرمج التطبيقي. أيضاً من الممكن رزم الصفات غير الهندسية والسيطرة عليها عند مستوى محطة العمل، كما سنشرحها في البند القادم.

2.7.3 صفات مميزة لمتعدد الخطوط (Polyline Attributes)

متعدد الخطوط له ثلاثة صفات هي:

1- شكل الخط (Line Style)

2- لون الخط (Line Colour)

3- عرض الخط (Line Width).

كما في الشكل 20.3، وهي صفات غير هندسية. يتم تثبيت الصفات الفردية لمتعدد الخطوط بواسطة إجراءات الصفة:

```
void' gset – linewidth (linewidth)
void gset – linetype (linetype)
void gset-line- colr-ind (line-colr-ind)
```

الشكل 20.3 شكل وعرض متعدد الخطوط

تقوم هذه الإجراءات بإعداد معامل قياس عرض الخط، نوع الخط ومؤشر أو دليل لون متعدد الخطوط (Polyline Colour Index) على التوالي.

يتم حساب معامل قياس عرض الخط نسبة إلى عرض قياسي لكل محطة عمل.

يعتبر معامل قياس 1.0 هو بديل افتراضي للخط الذي سيقوم بإخراجه على محطة عمل حقيقية. تكون المعلمية linewidth هي مضاعفات سمك البديل الافتراضي للخط. لا يمكن البرنامج أن يعلم بدون استخدام دالة استعلامية (Inquiry Function) فيما إذا كان بمقدور الجهاز الحقيقي أن ينتج خطوطاً لها سمك حسب الطلب. لهذا علينا أن ندرك

ذلك، بالرغم أننا يمكن إلى حد ما تحديد قيمة لصفة مطلوبة، لكـــن المحـــددات الماديـــة للعارضة سوف تملي علينا الحصول على أحسن مطابقة مع متطلباتنا.

يتم تحديد أنواع الخط بواسطة المؤشر linetype في إجراء gset-linetype. يكـــون البديل الافتراضي هو خط متصل (Solid Line). المؤشرات الأخرى تعطي أنواع مختلفـــة للخط المتقطع والمنقط. يتم تحديد لون الخط بواسطة دليل أو مؤشر في جدول الألـــوان. يستخدم GKS منظومة الثلاثة ألوان الأحمر- الأخضر- الأزرق (RGB) حيث يعمـــل بطريقة تقريباً مماثلة إلى عمل معظم أجهزة المراقبة الملونـــة (Colour Monitors). يتـــم تكوين الألوان بمزج الألوان الأساسية، أحمر وأخضر وأزرق بكميات قابلة للضبط. هكـــذا إذن يحتوي جدول الألوان على ثلاثة أعمدة كما مبين في الشكل 21.3.

Index	R	G	B
0	0.0	0.0	0.0
1	1.0	1.0	1.0
2	1.0	0.5	0.5
.	.	.	.
.	.	.	.

الشكل 21.3 جدول الألوان

يقوم دليل اللون (Colour Index) باختيار صف من الجـــدول والمداخـــل الثلاثـــة للأعمدة تعطي كمية الألوان من الأحمر والأخضر والأزرق المطابقة لهذا الدليل أو المؤشـــر. لذلك نستطيع تحديد الألوان بأسلوب مستقل عن الجهاز، وتتكون مداخل الجداول مـــن أرقام حقيقية ما بين 0.0 , 1.0، حيث تعني 0.0 لا وجود لهذا اللون الأساسي، وأمـــا 1.0 تعني الكمية المستخدمة هي الحد الأقصى من اللون الأساسي. سوف نقوم ببحث موضوع الألوان وأنظمتها أكثر تفصيلاً في الفصل السابع.

تقوم محطة العمل الحقيقية بتفسير قيم الألوان RGB المحددة بأحسن ما يمكن. علـــى سبيل المثال، عارضه ملونة RGB بسيطة قد تتيح للمستفيد فقط الألوان الناتجـــة عندمـــا تكون الألوان الأساسية إما كليا مفتوحة (fully on) أو كليا مغلقة (fully off). مثل هـــذه العارضات تستطيع إنتاج ثمانية ألوان كما يلي:

1. الألوان الأحمر، والأخضر، والأزرق (عندما يكون أحد الألوان مفتوحاً كليـــاً On واللونين الآخرين مغلقة كلياً Off).

2- اللون الأسود (كل الألوان مغلقة)، واللون الأبيض (كل الألوان مفتوحة).

3- اللون الأزرق الداكن (الأزرق والأخضر مفتوحتين، والأحمر مغلق) Magenta

4- اللون الأحمر مزرق (الأزرق والأحمر مفتوحين والأخضر مغلق)Cyan

5- اللون الأصفر (الأحمر والأخضر مفتوحين والأزرق مغلق).

في مثل هذه المنظومة، أي قيمة في الجدول أكبر من 0.5 تسبب أن يكون اللون الأساسي مفتوح كلياً، بينما قيمة أقل من 0.5 سيسبب أن يكون اللون الأساسي مغلق.

يتم تثبيت قيم مداخل جدول الألوان بقيم غير افتراضية على محطة عمل منطقية باستخدام الرمز التعريفي ws-id بواسطة الإجراء "إعداد تمثيل الألوان" set colour representation) كما يلي: void gset – colr-rep (ws-id, colr-ind, colr-rep)

حيث يشير الدليل colr-ind إلى صف في الجدول. يتم تمرير الكمية المطلوبة من الألوان الأحمر، والأخضر والأزرق من خلال تركيب رزمة الألوان (color-bundle) التالية:

```
typedef struct
{
    Gfloat x; /*red */
    Gfloat y; / * green*/
    Gfloat z;   /*blue*/
} Gcolr-rep;
```

حيث أن z,y,x تحدد الشدة المعيارية (Normalized Intensities) للألوان الأحمر والأخضر والأزرق على التوالي . هكذا نستطيع تعريف اللونين الأخضر والأزرق الداكن على محطة عمل مسماة CRT كما يلي:

```
#define GREEN  1
#define MAGENTA  2
:
static Gcolr – rep green =  {0.0, 1.0, 0.0}  ;
static Gcolr-rep magenta = { 1.0 , 0.0, 1.0 } ;

gset – colr- ind (CRT, GREEN, &green);
gset – colr – ind (CRT, MAGENTA,&magenta);
```

لاحظ ذلك، نستطيع استخدام نفس قيمة الدليل للإشارة إلى ألوان مختلفة على محطات عمل مختلفة. وبالتالي يمكن عرض نفس الكيان الأولي وبألوان مختلفة على محطتي عمل ملونة.

3.7.3 الصفات المميزة للنص (Text Attributes)

أصبحت الصفات المميزة للنص ذات أهمية كبرى بسبب الطريقة الحديثة في إنتاج نصوص عالية النوعية في صناعة الطباعة حيث اعتمدت رؤية كل رمز ككيان مستقل للرسم.

توفر المنظومات الحديثة للرسومات صفات مختلفة للنص حيث تتيح للمستفيد أن يبني نوعية عالية من عروض النص. تشمل هذه الصفات على:

1- حجم الرموز (Size).

2- نسبة مربع أقصى(Aspect Ratio) أي نسبة عرض الرمز إلى طوله.

3- لون الرمز.

4- اتجاه النص المنتج على العارضة.

بالنسبة للتطبيقات التي يتطلب فيها التعامل الدقيق مع النص، يمكننا التحكم بالفراغات بين الرموز واختيار أشكال وطواقم مختلفة لحروف الطباعة.

عند هذه المرحلة، إذ كنت متلهفاً للتواصل مع جوانب أخرى للرسومات بالحاسوب، ما عليك إلا أن تترك ما تبقى من هذا البند. ليس هناك فقط ثمانية إجراءات في منظومة GKS لتثبيت الصفات المميزة للنص، ولكن هنالك أيضاً عدة أنواع جديدة من البيانات. إذا كنت ترغب عند هذه المرحلة كتابة برامج تحتوي على الأكثر جمل بسيطة (نصوص ذات ارتفاعات ثابتة من اليسار إلى اليمين)، قد تكون القيم الافتراضية المعدة (Default Settings) مرضيه.

تكون الإجراءات في منظومة GKS لأعداد الصفات المميزة كما يلي:

```
void gset – char – ht (char-ht)
void gset – expan (expan)
void gset – space (space)
void gset – text – colr – ind (colr-ind)
void gset – char- up –vec (char-up-vec)
void gset-text-path (text-path)
void gset-text-align (text-align)
void gset – fontprec (fontprec)
```

ما يلي هو شرحاً لهذه الإجراءات:

1- نقوم بتحديد حجم الرموز كما يلي:

أ. تثبيت ارتفاع الرمز بواسطة char – ht في منظومة إحداثيات WC في إجراء gset-char-ht.

ب. اختيار معامل تقييس (Scaling Factor) "expan" في إجراء توسيع الرموز gset-char-expan كما مبين في الشكل 22.3

Normal

Large

Narrow

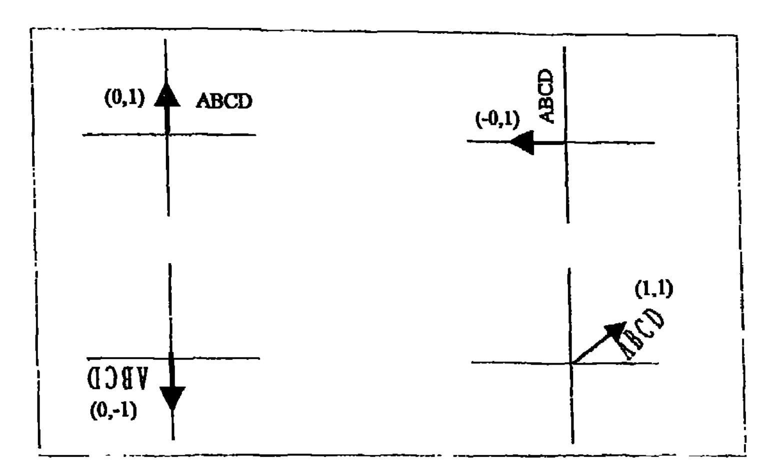
Fat

الشكل 22.3 إعداد حجم الحرف

ج. نستطيع توفير فراغات إضافية بين الرموز بواسطة الإجراء gset-space.

2- يمكن اختيار لون صف رموز النص من خلال التأشير في جدول الألوان بواسطة الإجراء gset-text-colr-ind. لقد تم استخدام نفس جدول الألوان في حالة ألوان متعدد الخطوط، والذي تم تثبيته من خلال الإجراء gset-colr-rep. بالرغم من أننا استخدمنا جدول واحد، لكن يمكننا استخدام مؤشرات مختلفة للنص ولمتعدد الخطوط. هكذا يمكن استخدام أكثر من لون واحد في نفس الوقت.

3- تكون الطريقة الاعتيادية في وضع نص على صفحة، وعلى الأخص النثر الإنكليزي، هو من اليسار إلى اليمين. مع ذلك، حتى على أبسط مخطط بياني (graph) ، عادة نرغب أن تكون لدينا القدرة. بوضع العلامات والعناوين في اتجاهات أخرى، كأن تكون موازية للمحور الصادي (y-axis). يتم التحكم بالاتجاه من خلال متجه (voctor) يدعى متجه الرمز للأعلى (character- up-vector) حيث يتم تثبيته بواسطة الدالة : gset – char-up-vec. بعض الاختيارات لهذه المتجهة مبينة في الشكل 23.3.



الشكل 23.3 متجه الحرف للأعلى

4- أيضاً قد نرغب في التحكم بالمسار الذي تسلكه الرموز حالما يتعين الاتجاه كما هو موضح في الشكل 24.3.

P
U
tfel right
d
o
w
n

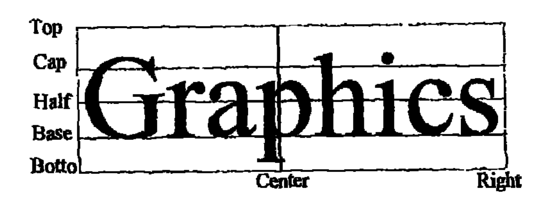
الشكل 24.3 مسار النص

والذي يمكن تحقيقه بواسطة تثبيت مسار النص في الإجراء gset-text-path. نوع البيان Gtext-part يكون من النوع القابل للعدد (Enumerated) والذي يمكن أن يتأخذ القيم التالية: GRIGHT-PATH, GLEFT-PATH, GUP-PATH, GDOWN-PATH.

5- نقوم بتحديد الموقع في دالة النص. في مثالنا البسيط، لقد استخدم البديل الافتراضي لهذا الموقع، والذي تم افتراضه ليكون الزاوية السفلى لليسار لصف من الرموز (

في فضاء WC). يكون اختيار هذا التراصف(Alignment) طبيعي في كثير من التطبيقـات، لكن يجعل دقة تعيين مكان النص أمراً صعباً إلى حد ما. على سبيل المثال، لـو أردنـا أن نمركز نص عند نقطة معينة في فضاء WC قد نحتاج إلى حساب مكان هذه النقطة . هـذه الحسابات إن كانت ضرورية قد نحتاج إلى معرفة حجم الرموز لربما لكل محطـة عمـل وعدد الفراغات بين الرموز. إذن بدلاً من أن يقوم برنامج التطبيق بإجراء هـذه العمليـة الحسابية المربكة يمكننا استخدام إجراء تراصف النص (text – alignment) وهو: gset-text-align.

إن عملية اختيار أسلوب التراصف يمكن إنجازه بصورة مستقلة بالنسـبة للاتجـاهين y,x. يتم تحديد التراصف بواسطة مؤشرات التي تعكس خواص صف الرموز. كما يمكننـا مشاهدته في الشكل 25.3 . حيث نجد هنالك أربعة اختيارات للتراصف المتوازي وسـتة للتراصف العمودي.



الشكل 25.3 ترصيف النص

يوجد هنالك عدد أكبر من الاختيارات للتراصف العمودي، حيث ينبغي أن نـأخذ بنظر الاعتبار الحروف الكبيرة (الاستهلالية Uppercase) والحـروف الصغـيرة (غـير استهلالية – Lowercase).

يحتوي Gtext- align على زوج من أنواع البيانات القابلة للعد التي تحدد الـتراصف المطلوب. على سبيل المثال، نستطيع مركزة صف من الرموز من خلال الدالة التالية:

static text-align = {GCENTRE-HOR, GHALF-VERT} ;

6- يتيح لنا الإجراء الأخير gset-fontprec في اختيار طاقم حروف الطباعة وانتقـاء الدقة التي نرغب أن تستخدمها المكونات المادية لإنتاج النـص. يتكـون نـوع البيـان Gfontprec من مركبتين:

أ. طاقم حروف الطباعة المطلوب ويمثل برقم صحيح.

ب. الدقة المطلوبة وتمثل بنوع من البيانات القابلة للعد (Enumerated).

سوف نقوم ببحث دقة النص في الفصل السادس لعلاقته بقضايا التنفيذ. إن المشكلة الأساسية تدور حول الصعوبات في إنتاج نص ذو نوعية عالية. تحتاج النوعية العالية من المخرجات مكونات مادية وبرمجيات متطورة. بالإضافة، يتوجب من المنظومة القيام بمزيد من العمل لإنتاج رموز ذات صفات محددة. لضمان وضع الرموز بصورتها الصحيحة بواسطة عمليات الرسومات كالتحويلات تضيف بعد آخر لتعقيد المشكلة. من خلال تحديد الدقة المطلوبة، يعني هذا الطلب من المنظومة أن تعمل بكل طاقتها حسب الضرورة لتوليد مخرجات عالية النوعية. هنالك ثلاثة مستويات لدقة النص معرفة هنا:

أ. صف من الرموز String،

ب. رمز Character،

ج. ضربة أو شوط Stroke.

تعتبر دقة صف من الرموز هي أدنى مستوى للدقة وبالتالي نحصل على أسرع إنتاج لمخرجات النص . وأما دقة الضربة تعتبر أعلى مستوى للدقة وبناء على ذلك يكون أبطأ وأكثر صعوبة في التنفيذ.

نشأت قضايا تنفيذية عديدة كنتيجة لهذه القدرات في تحديد عروض نص دقيقة وبأنواع مختلفة. كما كانت الحالة مع صفات متعدد الخطوط كاللون وسمك الخط، حيث أن تحديد الصفات المرغوبة في برنامج المستفيد لا يعطي ضماناً بقدرة المكونات المادية من إنتاج كيانات أولية معروضة حسب الطلب. بالنسبة للصفات المميزة كلون النص، سنرى أحسن تطابق للصفة المطلوبة. لكن بالنسبة للصفات المميزة الأخرى، ما نقصده بأحسن تطابق لربما يعتمد على التنفيذ. من المحتمل تحديد النص بدقة في برنامج، لكن بعد حين نجد أن المنظومة الحقيقية لا تتمكن من إنتاج هذه النوعية ولربما لا تنتج أي نص للإخراج على الإطلاق وعدم استطاعتها في تناول الطلب . من أجل تجنب مثل هذا الحدث قد نحتاج استخدام دليل المستفيد وإجراءات استعلامية بعناية.

4.7.3 صفات مرزومة (Bundled Attributes)

يستطيع البرنامج تثبيت الصفات غير الهندسية إما بصورة انفرادية، كما سبق شـرحه و من خلال الرزم (Bundles). في الآلية الأولى، يتم تثبيت كل صفـة إلى قيمـة غـير إفتراضية (Nondefault Value) بواسطة استدعاء إجراء بصورة منفصلة وسيكون للكيـان الأولي نفس الصفات على كل محطة عمل باستثناء لونه. أما الصفـات المرزومـة يمكـن إعدادها في مجموعات حيث توفر هذه آلية ذات مرونة كبيرة. يمكن استخدام الصفـات المرزومة لجعل ظهور الكيانات الأولية على محطات عمل مختلفة وبطرق مختلفة.

يمكننا إدراك أهمية هذه الآلية من المثال التالي. تأمل مثالنا السابق حيث فيـه لـدى المستفيد عارضه CRT ملونة وراسم قلمي. تستطيع العارضة CRT إنتاج على الأقل بعض الألوان ولكن الراسم القلمي يستطيع عرض خطوط سوداء على صفحة ورقة بيضاء فقط.

افترض نحن نرغب أن تكون لدينا القدرة على التمييز بين شكلين مختلفين للخطـوط. على العارضة الملونة، عادة يتم التمييز بين الخطوط بواسطة الألوان. إذا استخدمنا نفـس الآلية، أفضل ما يمكن أن يعمله الراسم القلمي هو عرض كلا الخطين بـاللون الأسـود وبالتالي لا يمكن التمييز بينهما. لربما يكون البديل هو جعل أحد الخطين متصـل (Solid) والآخر متقطع (Dashed). مع أن هذا لربما يحل مشكلة الراسم القلمي، لكن المستفيد قـد يرى الخطوط المتقطعة الملونة أقل إغراءً على العارضة CRT. في الحقيقة ما نريد توفره هـو أن تكون لدينا ألوان مختلفة على CRT، وأنواع مختلفة من الخطوط على الراسم القلمـي. لذا الصفات المرزومة تحقق لنا هذه الرغبة.

مع كل صفة غير هندسية توجد هناك إشارة إعلام (Flag) تدعى إشارة إعلام منبـع الهيئة (Aspect Source Flag) التي تحدد فيما إذا كانت الصفـة مرزومـة أو انفراديـة. الجميع تم إخبارهم بأن هنالك ثلاثة عشر صفة من هذه الصفات وتشكل إشارات إعـلام مصفوفة.

الثلاث الأولى من إشارات الإعلام تسيطر على صفات متعدد الخطوط: شكل الخط، معامل مقياس عرض الخط واللون. نقوم بإعداد مؤشر (index) لكل محطة عمل ولكـل

مجموعة من الصفات المرزومة لمتعدد الخطوط من خلال الإجراء "ثبت تمثيل متعدد الخطوط".

```
voide gset-line-rep (ws-id, line-ind, line-rep)

Gint ws-id, line-ind;
Gline-rep *line-rep;
typedef struct
{
      Gint type;
      Gfloat width;
      Gint colr;
} Gline- rep;
```

هكذا نستطيع تكوين خطوط حمراء وخضراء متصلة على العارضة CRT مع مؤشرين واحد واثنين، بينما خطوط ممتلئة ومتقطعة على الراسم القلمي يمكن تكوينها مع نفس المؤشرين كما مبين في الشكل 26.3.

Index	WS	Type	Width	Color	
1	CRT	______	1	•——▶	(1., 0., 0.)
1	Plotter	______	1	•——▶	(0., 0., 0.)
2	CRT	______	1	•——▶	(0., 1., 0.)
2	Plotter	_ _ _	1	•——▶	(0., 0., 0.)

الشكل 26.3 جدول الرزمة

يتم تغيير الصفات ضمن البرنامج بواسطة الإجراء "ثبت مؤشر متعدد الخطوط" (set polyline index)

```
void gset – line-ind (line-ind)
Gint line – ind;
```

توجد هنالك إجراءات مشابهة للكيانات الأولية الأخرى (مثل ثبت تمثيل النص وثبت مؤشر النص).

سوف نفترض أن ، جميع الصفات تثبت بشكل انفرادي، إما باستخدام البديل الافتراضي أو في مرحلة التمهيد. مع أن استخدام الصفات المرزومة لا يوفر فقط مزيداً من المرونة للمبرمج، ولكن أيضاً يعطي برامج مرتبة وذلك تجنبنا من تكرار سلسلة طويلة من عمليات تثبيت الصفات كلما أردنا تبديل شكل الخط أو النص. نستطيع وضع جداول

الرزم مرة واحد كجزء من مرحلة التمهيد. هكذا يمكن المناورة بالصفات في برنامج وذلك عبر عدد قليل من المؤشرات.

8.3 كيانات أولية أخرى (Other Primitives)

بالرغم من أننا نستطيع التأمل تقريباً في عدد كبير جداً من كيانات أولية إضافية مختلفة، تتراوح ما بين أنواع مختلفة من المنحنيات إلى مجموعات من الأشكال، لكن معظم المنظومات توفر مجموعة صغيرة من هذه الكيانات التي يتم إسنادها في جميع التطبيقات.
توفر منظومة GKS ثلاثة كيانات أولية إضافية إلى متعدد الخطوط والنص هي:

1- متعدد العلامات Polymarkers.
2- مساحات ملء Fill Areas
3- صفوف خلية Cell Arrays

يمكن توليد كيانات أولية أكثر تطوراً من خلال هذه المجموعة، لربما بواسطة استخدام طرق تقريبية. على سبيل المثال، قد نستطيع مطابقة (Fit) منحنى بواسطة متعدد خطوط. وكذلك قد نجد مصائد للأخطاء في منظومة معينة ، كمثل رسم كيان أولي عام الذي سنقوم ببحثه لاحقاً في هذا البند. لقد تم التأكد من أن هذه الكيانات الأولية وافية في معظم التطبيقات ثنائية الأبعاد.

في الفصل الثامن، سنرى كيف يمكننا توسيع هذه الكيانات الأولية ثنائية الأبعاد التي تم دراستها هنا إلى مجموعة كيانات أولية ثلاثية الأبعاد.

1.8.3 متعدد العلامات (Polymarker)

العلامات عبارة عن رموز، كالنجوم (Starts) واصلبه (Crosses) ونقاط (Dots) وعلامات الزائد (Pluses)، غالباً تستخدم في رسم المخططات البيانية. بالرغم من محاولتنا استخدام رموز فردية في أوامر النصوص (Text Commands)، لكن هذه الطريقة وجدت غير عملية. مثال ذلك، لو رغبنا تغيير مخطط بيانات الذي يستخدم فيه خطوط تصل بين نقاط البيانات من خلال متعدد الخطوط إلى مخطط بيانات يستخدم فيه سلسلة من النجوم

لتأشير النقاط، حيث لا نريد أن يكون هذا التبديل في المخطط أن يؤدي إلى تغيير هـــام في البرنامج. يكون الحل هو استخدام دالة متعدد العلامات (polymarker function):

```
void gpolymarker (num-pt, points)
Gint num – pt;
Gpt *points;
```

كما هو الحال مع متعدد الخطوط، النقاط points هي عبارة عن مصفوفـــة عـــدد نقاطها num-pt. يتم وضع العلامة المستخدمة حالياً عند كل نقطة في المصفوفة، حيـــث تكون إحداثياتها في فضاء WC. يتم تكوين رمز العلامة الحالية بواسطة الإجراء "ثبت نوع العلامة" (set marker type):

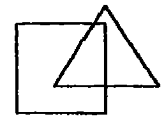
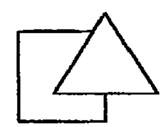
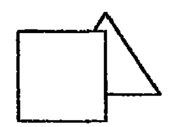
```
void gset – marker-type (marker-type)
Gint marker-type;
```

حيث يكون marker-type مؤشر في جدول العلامات.

إضافة إلى نوع العلامة، متعدد العلامات له مجموعة من صفات اللـــون وصفـــات معامل قياس حجم العلامة (marker scale-factor attribute).

2.8.3 مساحة الملء (The Fill Area)

في منظومة GKS مساحات الملء هي عبارة عن مساحات مضلعـــة (Polygonal Areas). نستطيع تعريف المضلع بواسطة قائمة مرتبة حسب النقـــاط أو رؤوس المضلـــع (Vertices). يتم توصيل الرؤوس المتتالية بقطع من الخطوط تسمى الحافـــات (Edges). يوصل الرأس الأخير بالرأس الأول لتتكون لدينا حدود مغلقة. من هذا قد يتضـــح أنـــه باستطاعتنا تمثيل مضلع له n من النقاط كمتعدد خطوط له n +1 من النقاط وذلك بإضافة نقطة إضافية إلى متعدد الخطوط يكون موقعها نفس مكان النقطة الأولى. بهـــذه الطريقـــة يمكننا رسم حدود المضلع. يستخدم الاسم مساحة ملء (fill area) بدلاً من مضلع وذلك للتأكيد بأن مساحة ملء لها مساحة داخلية (Interior) في حين المضلع المكون من متعـــدد خطوط قد لا يحتوي على مساحة داخلية. نستطيع تلوين داخل مساحة المـــلء أو ملئـــه بأشكال مخططة. يعتبر اختيار أشكال الملء (Fill Styles) صفة الكيان ا لأولي لمســـاحة الملء. الشكل 27.3.

الشكل 27.3 مضلع مقابل متعدد خطوط

يوضح لنا الفارق المهم بين مساحة الملء ومتعدد خطوط مغلق. لو عرفنا اثنين من متعددات الخطوط المثلث والمربع نحصل على الصورة الموضحة في الجهة اليسرى من الشكل 27.3. لكن لو استخدمنا مساحات ملء نستطيع الحصول أما على الصورة في الوسط أو الصورة المبينة على جهة اليمين معتمدة على ترتيب (أو أسبقية) وضع المثلث أو المربع على العارضة.

يتم تعريف مساحة ملء بواسطة الإجراء:

```
void gfill – area (num-pt, points)
Gint num-pt;
Gpt *points;
```

حيث تكون النقاط points عبارة عن مصفوفة تحتوي على num-pt من الرؤوس. منظومة GKS تعرف أربعة أشكال أو طرازات (Styles) داخلية لمساحة الملء هي:

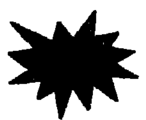
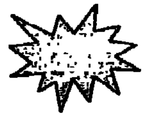
1- بجوف Hollow،

2- صماء (غير بجوف) Soild،

3- مضلل Hatch،

4- مخطط Pattern.

هذه الأشكال موضحة في الشكل 28.3 .

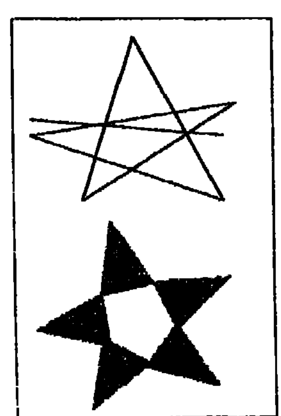





الشكل 28.3 أشكال داخلية لمساحة الملء

يتم اختيار الطراز أو الشكل (Style) بواسطة الإجراء "ثبت الطراز الداخلي لمساحة الملء" (set fill area interior style). يكون انتقاء النموذج الخاص أو اللون (لإملاءات صماء أو غير مجوفه) بواسطة الإجراء "ثبت مؤشر طراز مساحة الملء" (set fill area style index).

في الواقع ملء منطقة داخلية معرفة بواسطة مجموعة رؤوس أو حافات إما بلون كامل أو مخطط يستحيل تنفيذه إلا على منظومة المسح الشبكي (Raster-System). الحصول على إملاءات مجوفة (Hollow Fills) بواسطة رسم حافات مساحة الملء وبدون الاهتمام حول المنطقة الداخلية وبالتالي يمكن عرض مساحات الملء على منظومات المسح العشوائي (Random-Scan System). أيضاً نستطيع إنتاج إملاءات مضللة (Hatch Fills) على منظومات المسح العشوائي.

تقوم منظومة GKS بتعريف الحافة أو حدود مساحة الملء كونها تقع داخل المنطقة. بالأحرى هذا الاختيار يكون كيفي (Arbitrary) ولا يسمح لنا من تكوين مساحة كلها حمراء وحافاتها خضراء باستثناء استخدام كيانين أوليين هما: مساحة ملء خضراء ومتعدد خطوط حمراء. هناك منظومات أخرى، كمنظومة PHIGS، التي تتيح للمستفيد أن يعمل مع حافة مساحة ا لملء بمعزل عن المنطقة الداخلية لمساحة الملء.



الشكل 3 29 الملء بواسطة خطوط المسح

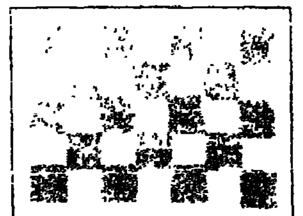
هنالك عدة طرق يمكن تحديد مسألة ماذا سيكون داخل أو خارج مساحة الملء. تستخدم منظومة GKS طريقة خط المسح (Scan – Line) . لنأخذ بنظر الاعتبار مساحة الملء المبينة في أعلى الشكل 29.3.

تقوم منظومات المسح الشبكي بعرض محتويات مخازنها الانتقالية للصورة (Frame Buffer) خط بعد خط. كل خط من هذه الخطوط المتوازية تدعى بـ "خط المسح – Scan-Line" . أثناء تتبعنا خط المسح عبر مساحة الملء، يمكن اعتبار نقطة تقع داخل مساحة الملء إذا كان عدد مرات تقاطع هذا الخط مع الحافات هو عدد فردي قبل الوصول إلى النقطة. (ملاحظة: طالما خط المسح لا

يتقاطع مع أي من الرؤوس ينطبق هذا التعريف بصورة متساوية إذا اعتبرنا الخط يمر عبر المضلع بأي اتجاه). على أساس هذا التعريف، لقد ملئت مساحة الملء كما مبين في أسفل الشكل 29.3. سنقوم بشرح تفاصيل خوارزميات الملء في الفصل السابع.

3.8.3 صفوف خلية (Cell Arrays)

صفوف خلية عبارة عن مساحة ملء مستطيلة حيث يستطيع المستفيد من تعريف نموذج أو نمط ملء (Fill Pattern). يتم التداول معها بواسطة الإجراء gcell-array. يعرف هذا الكيان الأولي مساحة مستطيلة في فضاء WC مقسمة إلى خلايا كما مبين في الشكل 30.3.



الشكل 30.3 صفوف خلية

يتم ملء كل خلية بلون منتخب بواسطة مؤشر يتم الحصول عليه وذلك بالبحث من خلال مصفوفة محددة في قائمة المعلميات.

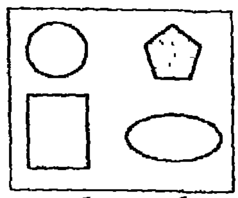
كان الغرض من مصفوفة الخلية هو تجهيز المستفيد بكيان أولي مشابهة للمسح الشبكي (Raster – Like Primitive). بالرغم من تعريف مصفوفة الخلية في إحداثيات WC، كما هو مع جميع كيانات الإخراج في منظومة GKS، توجد هنالك بعض الصعوبات في التطبيق. منظومات المسح العشوائي قد لا تكون قادرة على إنتاج مخططات أو تشكيلات وفي عديد من المنظومات الشبكية، مخططات معرفة من قبل المستفيد قد تظهر غير متقنة ومثلمة (Ragged) أثناء التحويل إلى منظومة إحداثيات DC. لهذه الأسباب صفوف الخلية لم تثبت فائدتها في كثير من تطبيقات GKS.

4.8.3 كيانات أولية للرسم العام (Generalized Drawing Primitives)

يتم التداول مع الكيان الأولي للرسم العام من خلال الدالة التالية:

```
void ggdp (num – pt, points, gdp – id, gdp-data)

Gint num-pt;
Gpt *points;
Gint gdp-id;
Gdp-data *gdp-data;
```


الشكل 31.3 كيانات أولية للرسم العام

يتيح هذا الكيان الأولي الوصول إلى كيانات إخـــراج تحديـــداً، مولدة بواسطة المكونات المادية كالدوائر، والقطع النـــاقص (اهليـــج) والمستطيلات كما موضح في الشكل 31.3.

والتي لا تعرف ضمن منظومة GKS. بمـــا أن ليســـت كـــل المكونات المادية بإمكانها إنتاج جميع الكيانات الأولية حيث أن هـــذا الإجراء يكون معتمداً على الجهاز، كما هو الحال مع أي برنامج تستخدمه . تتيح معلميـات الاستدعاء مرونة كافية في إمكانية التداول مع عدد من أنواع الكيانات بواسطة ggdp.

كل كيان أولي متوفر للرسم العام له رمز تعريف gdp-id. وكذلك سيحتاج الكيـان الأولي الخاص في الرسم العام عدد من النقاط num-pt في المصفوفة points لتعريفه بصورة منفردة. على سبيل المثال، قد يكون num-id=1 دائرة و num-id=2 قد يمثل قوس مـــن دائرة. نستطيع تعريف دائرة بواسطة نقطتين – أحدهما مركز الدائرة والأخرى نقطة علـــى محيطها. أما القوس يحتاج إلى ثلاثة نقاط – مثال ذلك، مركز الدائرة ونقطتين على المحيـط. أن قيد البيانات (Data Record) المعرف للتنفيذ gdp-data قد يكون ضروريا لتحديـــد صفات الكيان الأولي، أو أي معلومة أخرى قد تحتاجه المكونات المادية . كبديل ، يمكـــن الحصول على صفات الكيان الأولي للرسم العام من صفات متعدد الخطـــوط أو صفـــات مساحة الملء. على سبيل المثال، قد تستخدم الدائرة صفات متعدد الخطوط الحالية لرســـم محيطها وصفات مساحة الملء لملأ مساحتها الداخلية. يمكن تحديد دائرة مركزها عند نقطة الأصل مع نصف قطرها واحد بواسطة عبارتين هما:

```
static Gpt circle – data [ ] = {{0.,0.}, {0. ,1.}}
ggdp (2, circle – data, CIRCLE- ID, NULL);
```

9.3 راسم بياني ذاتي التدرج *(A Self – Scaling Plotter)*

كمثال لعديد من السمات المميزة التي تم استعراضها في البند السابق، الآن ســـوف نقوم بتصميم دالة لراسم بياني ذاتي التدرج. ستقوم الدالة برسم منحنى واحد لعـــدد مـــن النقاط num-pt، تكون مخزونة في مصفوفة مشار إليها بواسطة xy-data. نحـــن نرغـــب بوضع علامات على المخطط ليشمل كلا المحورينx وy وعنوان في أعلى المخطط. لقـــد تم

الإشارة إلى هذه السلسلة أو مصفوفـات الرمـوز (Strings Or Character Arrays) بواسطة y-lable, x-lable و title على التوالي. إذن سوف يتم التداول مع هـذه الدالـة بواسطة البرنامج التالي:

```
Gint num- pt;
Gpt xy-data {NUM- POINTS};
Gchar *x-label, *y-label,*title;
        |
plot (num-pt, xy-data, x-label, y-label, title);
```

قبل البدء بكتابة هذا البرنامج دعنا ندرس بعض متطلبات هذا الإجراء. نظراً لأننـا نقوم بتمرير البيانات للإجراء فقط، لذا علينا تدريج مخطط البيانات بحيث يظـهر علـى مساحة العرض نفسها بغض النظر عن قيم البيانات. إذن ، ينبغي أن يكون إجرائنـا ذاتي التدريج أو التقييس (Self-Scaling). لربما نرغب في استخدام بعض الصفـات المميزة للنص، لأن عنونة المخطط قد تحتاج مركزتها، وإن أمكن توجيه نص عنوان المحور الصادي (y-axis) بحيث يكون على امتداد هذا المحور.

أيضاً قد نحاول إظهار الإخراج بشكل لائق وفيه ذوق بدلاً من أن يتكون فقط مـن متعدد خطوط للبيانات وخطوط للمحاور.

من أجل تدريج المخطط، علينا أن نجد الحد الأقصى (Maximum) والحـد الأدنى (Minimum) لبيانات النقاط . دعنا نفترض أن لدينا إجراء يدعى bounds الذي يقـوم بإيجاد وإعادة قيم الحد الأقصى والأدنى للبيانات. تكون كتابة هذا الإجراء بسيطة.

سبق وإن كتبنا إجرائين أحدهما للتمهيد init والآخر للإنهاء finish، اللذان نستطيع تنشيطهما هنا. هكذا، إذن سيتضمن إجرائنـا العبـارات المسـتعارة (Pseudecode) في الخطوات التالية:

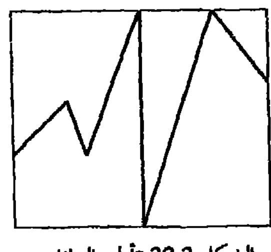
```
plot (num-pt, data, xlabel, ylabel, title)
Gint num-pt;
Gpt data [ ];
Gchar *xlabel, *ylabel, *title;
{
```

```
init (....);
bounds (....);
set-up – normalization- transforms (....);
draw – axes (....);
plot – data(....);
draw-labels- and - title (....);
finish(.....);
}
```

1.9.3 إعداد التحويلات المعيارية
(Setting up the Normalization Transformation)

تتوفر لدينا الآن برامج لأول خطوتين . يحتاج إعداد التحويلات المعيارية إلى تخطيط. سوف نستخدم تحويلين، أحدهما لمخطط البيانات والآخر للعلامات والعناوين. سنقوم بتكوين مخطط البيانات النهائي وذلك بوضع الجزئين معاً . إن استخدام تحويلين معيارين يضمن تطابق المخطط في نفس مساحة العارضة، مستقلا عن عدد النقاط ومدايات البيانات، في حين لا تزال الفرصة متاحة لنا أن نستخدم نفس حجم الرموز في العلامات والعنوان.

كل تحويل معياري يحتاج مواصفات النافذة وبوابة الرؤية المطابقة لها في فضاء NDC. يقوم الإجراء bounds بإعادة أربعة قيم وهي xmin, xmax, ymin, ymax .



الشكل 32.3 تأطير البيانات

تشكل النقاط (x_{min}, y_{min}) و (x_{max}, y_{max}) إحداثيات الزاوية السفلى لليسار والزاوية العليا لليمين للمستطيل الذي تنطبق فيه جميع نقاط البيانات (Data Points). يدعى هذا المستطيل أما إطار تحديد (Bounding Box) أو مدى البيانات (Extent of Data)، كما مبين في الشكل 32.3 .

الآن العبارات التالية:

```
Glim window1;
window1. x1 = xmin;
window1.y1 = ymin;
window1.x2 = xmax;
window1.y2 = ymax;
gset – win (1,&window1);
```

تقوم بإعداد نصف التحويل المعياري الأول: أي أن النافذة في فضاء WC هي إطار تحديد البيانات . بعد ذلك ، لإكمال هذا التحويل المعياري علينا أن نقرر أين سنضع هذه البيانات في فضاء NDC. هذا القرار سيعتمد على تحديد موقع العلامات والعنوان وكم سيكون حجم الرموز. سنقوم بجعل حجم الرمز 5 بالمائة من حجم الشاشة، ونحاول مركزتها في مساحات مناسبة على العارضة. ولو تركنا 10 بالمائة من فضاء NDC منطقة حرة في الأسفل بالنسبة لعلامة محور-x و 10 بالمائة منطقة حرة بالنسبة لعلامة محور-y، هذا سيضمن عدم تداخل البيانات مع النص. كما مبين في الشكل 33.3.

y Label Area	Ttile Area
	Plot Area
	x Label Area

الشكل33.3 مخطط العارضة لبرنامج الراسم البياني

لو تركنا 10 بالمائة من الجزء العلوي في فضاء NDC منطقة حرة حيث تكون لدينا مساحة كافية لوضع العنوان. الآن يمكننا تعريف بوابة رؤية مناسبة للبيانات وكما يلي:

```
Glim viewport1;
viewport1. x1= 0.1;
viewport1. y1=0.1;
viewport1. x2 = 1.0;
viewport1.y2 = 0.9;
gset – vp (1, &viewport1);
```

سوف يستخدم التحويل المعياري الثاني لكلا علامات المحور والعنوان . بما أن الرموز يمكن أن تظهر بأي حجم في فضاء WC، لذا باستطاعتنا انتقاء أي من الوحدات التي نرغب لهذا الجزء من المشكلة. يكون الاختيار الملائم لمثل هذه الحالات هو استخدام التحويل المعياري حيث تكون نافذته مربع وحدة (Unit Square) معرفة بالنقاط (0.0 , 0.0) و (1.0,1.0). مع هذا الاختيار تكون مواصفات ارتفاع الرمز (أو أي صفة هندسية) عبارة عن الجزء كسري من أبعاد النافذة، بما أن النص سيغطي كل الصورة كما مبين في الشكر 33.3 ، لذا تكون بوابة رؤيته المناسبة تشمل كل فضاء NDC. إذن تكون النافذة وبوابة الرؤية التي تم وصفها تواً هما البديل الافتراضي للتحويل المعياري (0).

يمكن رسم المحاور كمتعدد خطوط واحد ماراً من خلال النقاط (x_{min}, y_{max}) و (x_{min}, y_{min}) و (x_{max}, y_{min}). تم استخدام تراكيب البيانات (data) لاحتواء هذه القيم. من الممكن رسم مخطط البيانات وذلك باستدعاء متعدد الخطوط مرة واحدة:

```
gpolyline (num – pt, data);
```

كما هو الحال مع المحاور.

الآن نكون مستعدين لإضافة العلامات والعنوان. ستقوم هاتين باستخدام التحويل المعياري (0) الذي ينبغي اختياره بواسطة الإجراء gsel – norm-tran. كذلك تذكر أنه كان يتعين علينا استخدام gse1 – norm-tran لاختيار التحويل 1 قبل رسم المحاور والبيانات، وإلا سنبقى نستخدم البديل الافتراضي للتحويل. يكون البديل الافتراضي لاتجاه الرموز هو من اليسار إلى اليمين، لذلك أولاً سوف نضيف علامة المحور-x والعنوان في الأعلى. ينبغي تثبيت صفتين هما حجم الرموز وترصيف النص (alignment). وكذلك نستطيع اختيار شكل وطاقم رموز الطباعة المطلوبة. نظراً لكوننا اتخذنا قراراً بأن يكون حجم الرمز هو 5 بالمائة من النافذة والنافذة هي مربع وحدة، يتم تثبيت ارتفاع الرمز بواسطة

```
gset-char-ht (0.05);
```

نحن نرغب أن تكون صفوف الرموز في مركز المساحات المخصصة لها. سوف ندع منظومة GKS القيام بمهمة الترصيف واختيار مركز سلسلة الرموز أفقياً وعمودياً معاً بواسطة:

```
static Gtext-align center = {GCENTER-HOR, GHALF-VERT};
gset- text-align (&center);
```

يتم انتقاء طاقم رموز الطباعة من غير البديل الافتراضي بواسطة : gset – fontprec.

الآن نستطيع إضافة علامة محور – x والعنوان باستخدام gtext وذلك بإعطاء مركز الأطر المخصصة (Allocated Boxes) في WC:

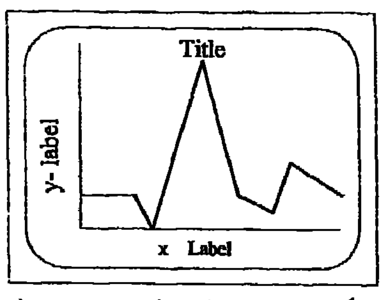
```
text – loc . x= 0.5;
text – loc .y = 0.05 ;
gtext (&text – loc, xlabel );
text – loc. x = 0.5;
text – loc. y = 0.95;
gtext ( &textloc, title);
```

نقوم بتوليد عنوان محور – y وذلك بتثبيت مسار النص على امتداد محور – y. نختار متجه نص أعلى (text up vector) مشيراً لليسار. لقد تم تثبيت الترصيف مسبقاً لذا نختار مركز المساحة المخصصة كموقع للنص:

```
char – up . x = -1.0;
char – up .y = 0.0;
gset – charup (&charup);
text- loc .x = 0.05;
text – loc.y = 0.5;
gtext (&textloc, ylabe ( );
```

لقد تم تكوين المخطط البياني كما شـرحناه وهو مبين في الشكل 34.3.

للبيانات البسيطة مع أن المخطط صحيح لكن شكل المخطط غير مثير للإعجاب. من أجل الوصول إلى فكرة تساعدنا في جعل المخطط يمتلك قليلاً من الذوق والجاذبية (Flashier)، دعنا نعمل جملة تعديلات بسيطة:



الشكل 34.3 مخرجات الراسم البياني : نسخة بسيطة

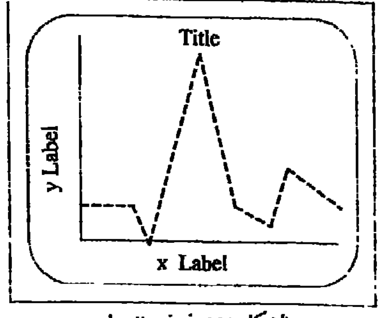
1- إضافة لون إلى الخط والمحاور بواسطة تثبيت ألوان في الجدول وباستخدام الدالة gset – colr-rep ومن ثم تغيير ألوان متعدد الخطوط بواسطة الدالة gset-line-colr-ind.

2- نستخدم متعدد خطوط غليظه من خلال الدالة gset – line-width للتميز بين المحاور والمنحنى.

3- نقوم بملأ المساحة تحت المنحنى وذلك بتشكيل مساحة ملء تكون حدودها من الأعلى المنحنى ومن الأسفل محور– x . يتم تشكيل مساحة الملء هذه وذلك بإضافة نقطتين (x_{max}, y_{min}) و (x_{min}, y_{min}) إلى تراكيب البيانات المدخلة. تقوم بملأ مساحة الملء مع نموذج التضليل المطلوب من خلال الدالة gset-fill-int-style والدالة set-fill-style-ind.

وأخيراً نستخدم متعدد الخطوط أيضاً للبيانات لتمييز قمة مساحة الملء.

الشكل 35.3 يبين المخطط الناتج للبرنامج الرئيسي البسيط بع الأخذ بنظر الاعتبار هذه التعديلات. البرنامج الكامل لهذا الإجراء أو الدالة موجود في الملحق C.



الشكل 35.3 نموذج مخرجات

10.3 ملفات ملحقة (Metafiles):

في هذا البند الأخير، سنتعرف على ملفات ملحقة كوسيلة خزن ونقل معلومات رسومات. قد تبدوا دراستنا للملفات الملحقة بعيدة عن مادة هذا الفصل، لكنها بالعكس وثيقة الصلة بالموضوع. في هذه المرحلة ينبغي عليك أن تكون مقتدراً في استخدام حزمة برمجيات الرسومات.

حالما نبدأ بكتابة برامج تطبيقات، قد ترغب إرسال نتائجك إلى زملائك إما لغرض استخدام بيانات شخص ما، أو لتوليد مخرجات على جهاز عالي النوعية، مثال ذلك مسجل أفلام الذي قد يكون غير متوفر موقعياً. هكذا إذن ، توفر لنا الملفات الملحقة طريقة لتحقيق هذه الأهداف. سنبحث طريقتين: إحداهما ضمن منظومة خاصة (في حالتنا منظومة GKS) والأخرى تلك التي تتيح لنا الاتصال بين منظومات الرسومات المختلفة.

تضمنت طريقتنا في تطوير البرامج التطبيقية في كتابة برنامج بدون معرفة الخواص المادية لأي من أجهزة الرسومات أو حتى نوع المعالج، ونقوم باستخدام هذا البرنامج كواسطة لنا في نقل معلومات الرسومات. ينبغي أن يعمل هذا الأسلوب ، إذا كان المتلقي لبرنامجنا مستخدماً نفس منظومة الرسومات أيضاً، ويكون لديه مترجم لغة C ورابط لغة (C Language Binding) بمنظومة GKS. لكن ربما لم يكن لديه كل هذا . بالإضافة قد لا تكون هناك استعدادات بإعادة ترجمة (Recompile) وتشغيل برنامج طويل. حيث تكون الاهتمامات موجهة فقط نحو معرفة المخرجات أو قد يرغب شمول أجزاء من هذه المخرجات في عروضه. إذا كان المستفيد يستخدم منظومة GKS، فإن ملف GKS الملحق (GKSM) يوفر طريقة إضافية في خزن وقراءة مخرجات منظومة GKS بصيغة يمكن خزنها في ملف أو نقلها. ومن الناحية الأخرى، يوفر الملف الملحق الرسومات (CGM) آلية ملف ملحق يمكن استخدامه مع تطبيقات تم تطويرها بواسطة أي حزمة برامج رسومات تقريباً.

1.10.3 ملف ملحق منظومة GKS (The GKS Metafile)

يمكن اعتبار كل البرامج التي تمت كتابتها بأنها تحتوي على نوعين من العبارات وهما: إجراءات منظومة GKS وأية عبارة أخرى. قد يبدوا هذا التعبير ساذج لكنه ذا أهمية كبيرة لو نظرنا إلى تنفيذ البرنامج ومن وجهة نظر محطة عمل منطقية. لنأخذ بنظر الاعتبار محطة عمل فعالة للإخراج فقط (output-only wokstation). حيث تقوم محطة العمل هذه بما يلي:

1- تستلم الأوامر لعرض كيانات أولية في منظومة GKS.

2- تستلم الأجزاء (Segments).

3- تغير الصفات المميزة.

4- قد يطلب من محطة عمل مسح سطح شاشتها.

كل ما تراه محطة العمل هو عبارة عن سلسلة من دوال GKS ولا ترى الأجزاء الأخرى من البرنامج التطبيقي التي تكون ضرورية في عملية توليد بيانات هذه الدوال.

كنموذج مفاهيمي مناسب لمحطة عمل GKS هو اعتبارها صندوق أسود أو مغلق (Black Box) الذي تكون مدخلاته دوال منظومة GKS ومعلماتها (Parameters). إذن، يمكن تجهيز محطة العمل هذه بقائمة دوال GKS وبيانات عبر وسائط أخرى غير البرنامج ومن المستطاع إنتاج نفس المخرجات. لذا يكون ملف GKS الملحق عبارة عن آلية كتابة وقراءة هذه المعلومات لتسهيل عملية نقل المعلومات.

الشكل 36.3 يوضح هيكل الملف الملحق.

header	item	item	....	item	end

الشكل 36.3 تركيب الملف الملحق

قد تتفاوت صيغة الملف الملحق من ثنائي (Binary) إلى شفرة ASCII ويتعلق هذا بقضايا التنفيذ التي سوف لا نقوم بشرحها . يتكون هذا الملف من عدد من القيود أو الفقرات (Items) . تحتوي فقرات المقدمة أو الصدارة (Header Items) معلومات حول الملف وصيغته . يكون من الضروري وجود فقرة إنهاء (Terminator Item) في معرفة نهاية الملف الملحق. بينهما توجد فقرات أخرى (كما مبين في الشكل 37.3).

item type	Data length	Data

الشكل 37.3 عناصر الملف الملحق

وكل فقرة من هذه الفقرات تمثل دالة في منظومة GKS كما تراها محطة العمل. الدالة تكون معرفة برقم وتتبعها بيانات الدالة.

إن محتوى ملف GKS الملحق هي مرآة لدوال الإخراج كما تراها محطة العمل. هكذا، إذن ، ملف ملحق للإخراج يكون مشابه إلى أي محطة عمل للإخراج . لهذا نقوم بفتح محطة عمل يكون نوعها مشابهاً إلى ملف ملحق للإخراج. حيث يخصص له رمز تعريف منطقي، كما هو الحال مع أي محطة عمل. الآن، طالما يكون الملف الملحق في حالة فعالة، تذهب المخرجات إليه، ومعه إلى أية محطة عمل فعالة أخرى، وستبقى هذه المخرجات في الملف الملحق.

نظراً لكون جميع المخرجات تذهب إلى هذا الملف الملحق (إن كان في حالة فعالة)، فإن ملف GKS الملحق يوفر أثر تتبع البرنامج. لنأخذ بنظر الاعتبار البرنامج التالي الذي تستخدم فيه دالة مسح شاشة محطة العمل (Clear Workstation):

```
init ( )

gpolyline (    );

g clear-ws (METAFILE);
finish(  );
```

حالاً قبل أن يتم إخماد (Deactivated) محطة عمل الملف الملحق (Metafile Workstation) وإغلاقه بواسطة الدالة finish، نستخدم الإجراء gclear- ws في مسح سطحه، لكن نظراً لكون محطة العمل الحقيقية هي عبارة عن ملف، لذا تكون الكيانات الأولية المتولدة بواسطة متعدد الخطوط في هذا الملف . عندما يتم غلق الملف الملحق، يكون محتواه جميع الكيانات الأولية التي تم عرضها فيه حينما كان فعالاً. بعدئذ تستطيع محطة عمل إدخال ملف ملحق (Metafile Input Workstation) من استخلاص الكيانات الأولية من هذا الملف الملحق الذي تم وضعه هناك قبل دالة مسح محطة العمل. بالرغم من أن العرض الأخير كان فراغاً، لكن كل الكيانات الأولية التي سبق عرضها تكون متوفرة في الملف الملحق هذا.

2.10.3 ترجمة ملف GKS الملحق (Interpreting a GKS Metafile)

نقوم بقراءة ملف GKS الملحق وذلك بفتح محطة عمل التي تكون من نوع الملف الملحق للإدخال (Metafile Input) ومن ثم معالجة الفقرات في الملف الملحق. إن جعل ملف ملحق للإدخال متوفر من خلال محطة عمل GKS اعتيادية يكون متوافقاً (Compatible) مع مفهومنا لمحطة العمل. مع ذلك، نحتاج إلى إجراءات خاصة لتناول فقرات الملف الحق، لأنها تقوم بتجهيز برابحنا بصيغ جديدة من المدخلات. "الإجراءات الثلاثة التي نحتاجها هي:

1- احصل على نوع الفقرة من GKSM (get item type from GKSM)

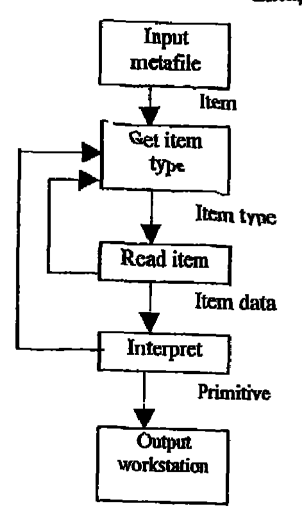
void gget – item- type (ws-id, itme –type, item – data-length)

2- إقراء الفقرة من GKSM (read item from GKSM)

void gread – item (ws-id, max-item-data-length, item-data)

3- ترجم الفقرة (interpret item)

```
void rinterpret- item( item-type, item-length, item-data)
Gint ws - id ;
Gint *item - type;
Gint *item- data - length;
Gint item- length;
Gint max - item - data - length;
Gitem- data *item data;
```



الشكل 38.3 ترجمة الملف الملحق GKS

تبدأ انسيابية عملية ترجمة أو تفسير الملف (الشكل 38.3) بفتح الملف الملحق مع دليل الملف (file pointer) مشيراً إلى أحد فقراته.

نقوم باستخدام الإجراء gget-item-type ليعلمنا ما هي هذه الفقرة هو طول قيد بياناته (Data Record). لاحظ ذلك، قيد البيانات قد يكون مختلفاً لنفس نوع الفقرة. على سبيل المثال، متعدد خطوط لثلاثة نقاط ومتعدد خطوط آخر لمائة نقطة يكونان من نفس النوع ولكن الثاني يحتاج إلى مساحة أكبر بكثير لقيد بياناته. دليل الملف يتم تقديمه ذاتياً إلى الفقرة

التالية. بعد تفحص نوع الفقرة، يمكننا إما أن نستخدمه أو ننتقل إلى الفقـــرة التاليـــة في الملف. عند قراءة فقرة يتم إعادة بياناته إلى البرنامج في تراكيب بيانـــات معتمـــداً علـــى التطبيق item – data. حيث تكون هذه البيانات متوفرة للاستخدام بواســـطة البرنـــامج التطبيقي.

نستطيع الحصول على فقرة من ملفنا الملحق وترجمته على جميع محطات العمل الفعالة من خلال الإجراء ginterpret-item. لو كانت الفقرة التي تم قراءتها هي متعدد خطـــوط وقررنا ترجمتها، فسيظهر متعدد الخطوط على جميع محطات العمل الفعالة للإخراج ومـــن ضمنها أي ملف ملحق فعال لمحطة عمل إخراج. هكذا، يستطيع برنامج أن يمر على الملف الملحق ويقرر ماذا سيفعل مع كل فقرة. أو يمكن للبرنامج أن يقرأ الملف الملحق بأكملـــه عند البداية، وبعدئذ يستمر كما يلي:

```
/ *  After Opening Metafile */

gget- item-type (META-IN, item-type, item – data- length)
while (* item- type ! = END- METAFILE)
{
     gread – item (META-IN, MAX- LENGTH, item – data)
     ginterpret – item (item- type, item –length, item – data)
     gget-item- type (META-IN, item- type, item – data- length)
}
/ * Continue*/
```

وبهذه الطريقة، نستطيع استخدام الملفات الملحقة لتجهيز وحدات بنـــاء (Building Blocks) للبرامج التطبيقية التي تكون مستقلة عن البرنامج المستخدم في توليد وحدات البناء.

3.10.3 ملف ملحق للرسومات الحاسوب (The Computer- Graphics Metafile)

نظراً لكون ملف GKS الملحق عبارة عن تتبع أثر تنفيذ برنامج في GKS، إذن مـــن الضروري أن يكون مترجم ملف ملحق له معرفة حول منظومة GKS. غالباً ما نرغب في استخدام ملف ملحق فقط لنقل مخرجاتنا إلى جهاز إخراج عالي النوعية كمسجل الأفـلام. مثل هذه الأجهزة قد تكون عالية الكلفة ولها قدرات محدودة في معالجة برنامج المســـتفيد الذي يستخدم منظومة رسومات خاصة مثل GKS. مع ذلك، مثل هذا الجهاز ســـيكون قادراً على توليد كيانات أولية للإخراج تستخدم في معظم المنظومات. عادة مثـــل هـــذه الأجهزة تعمل بصورة منفصلة عن الحاسوب (Offline) وينبغي جعلها متوفـــرة وقابلـــة

التداول من قبل فئة واسعة من المستفيدين، وليست مقتصرة علـــى الذيـــن يســـتخدمون منظومة GKS.

ما يحتاجه مثل هذا الجهاز كمدخلات تكون فقط معلومات ضرورية لإنتاج الصـورة مثال ذلك، قائمة بالكيانات الأولية وصفاتها المميزة. لقد تم اســـتحداث ملــف ملحــق للرسومات في الحاسوب [CGM86] كطريقة قياسية لتخزين هذا المعلومات. يكون هيكل الملف الملحق مشابهة إلى ملف GKS الملحق حيث يحتوي مقدمة ونهاية ومحتوى القيـــود. تكون الفقرات التي يتضمنها أكثر شمولية من الكيانات الأولية لمنظومـــة GKS . مـــع أن CGM يوفر وصف الصورة فقط وهو لا يوفر إمكانية برنامج متتبع لـ GKSM.

يمكن التوسع في استخدام الملفات الملحقة في الرسومات إلى حل مشاكل أخرى، وهـــي المشاكل الأكثر شمولية في التنقلية (Portability). أحياناً قد تحتاج بعض التطبيقات الـــتي تستخدم الرسومات أن تنقل معلومات أخرى غير معلومات الرسومات . على سبيل المثال:

1- لو قمنا بتصميم أجزاء ميكانيكية بواسطة برنامج CAD تفاعلي، قد يتضمـــن التصميم معلومات مثل المواد والتفاوت المسموح (Tolerances).
2- كذلك في مثالنا تصميم الدوائر الكهربائية قد يتضمن التصميم قيـــم العنـــاصر وأرقام الأجزاء (Part Numbers).
3- قاعدة البيانات ينبغي أن تحتوي على طريقة للتبادل، لكن ليست حصراً، علـــى معلومات الرسومات. إن قياســـي تبـــادل الرســـومات التمهيديـــة (Initial Graphics Exchange Standard – IGES) [IGES86] هـــو أحـــد هـــذه القياسيات. إن استحداث هذه القياسيات أصبح مجالاً لأنشطة كبيرة نظراً لنمـــو التطور المطرد للتطبيقات المستخدمة وحاجتنا إلى قواعد بيانات قياسية أكـــثر تطوراً.

تمارين

1.3 منظومة GKS لا تسمح لنا بإيقاف التقليم عند مستوى محطة العمل. اشرح لماذا يكون التقليم مطلوباً عند الانتقال من فضاء WC إلى NDC.

2.3 التطبيقات كبرنامج وضع مخطط للدوائر الكهربائية تستخدم نفس الأشياء المنظورة (مثال على ذلك مقاومات، ومكثفات) بصورة متكررة لتكوين صورة. بالنسبة لهذا المثال أو أي مثال آخر نجد من المهم تعريف مجموعة من الرموز (symbols) المناسبة. اكتب إجراءات لتوليد هذه الرموز باستخدام كيانات أولية في منظومة GKS. اجعل التحكم في حجم وموقع هذه الرموز جزء من معلميات الإجراء.

3.3 قد يكون برنامج رسم المخططات الذي تم تطويره في هذا الفصل بسيطاً وأقل مرونة مما نود أن يكون. باستخدام الاجراءات التي استعرضناها، أكتب إجراءً لرسم مخططات تتيح عمل ما يلي:

1- رسم مخططات متعددة.

2- طرق متعددة لعرض البيانات (مثل، مخططات قضيبية – Bar Charts ومخططات بأي – Pie Charts).

4.3 باستخدام منظومة GKS، استحدث مكتبة إجراءات لرسومات السلحفاة. ينبغي أن تكون هناك خمسة إجراءات أساسية وهي:

1- تمهيد موقع البداية للسلحفاه (place (x,y))

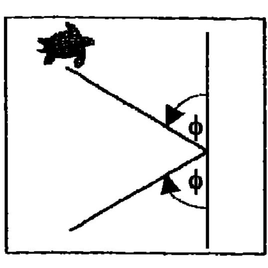
2- إلى الأمام forward (distance)

3- إلى اليمين right (angle)

4- إلى اليسار left (angle)

5- تحريك القلم إلى الأعلى أو الأسفل pen (up – down)

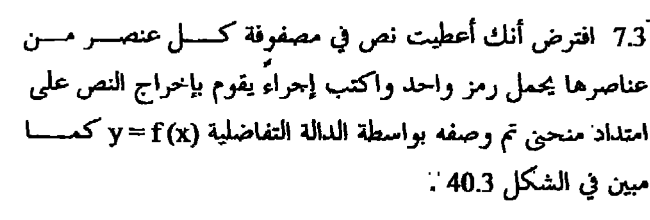
كيف يمكنك اختيار نافذة مناسبة ؟ كيف تتعامل مع استدعاء إجراء الـذي يحاول تحريك السلحفاة خارج النافذة؟



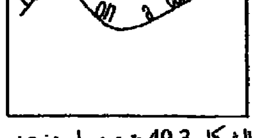
الشكل 39.3 السلحفاة

5.3 كطريقة مختلفة لتمرين 4.3 قد يكون ممتع خلــق انعكــاس للسلحفاة (Reflecting Turtle) هنا يتم وضـــع السـلحفاة في صندوق ثابت الحجم. أنها ترتد من الجوانب كالضوء الذي يرتــد في المرآة، كما مبين في الشكل 39.3.

6.3 معظم الناس وجدوا الصفات العديدة المميزة للنص هي أقــل السمات إغراءً في منظومة GKS. عرف مجموع أبسط لســمات النص، ومن ثم اكتب إجراءات لتنفيذها، مبتدأ مع إجراءات منظومة GKS.



7.3 افترض أنك أعطيت نص في مصفوفة كــل عنصــر مــن عناصرها يحمل رمز واحد واكتب إجراء يقوم بإخراج النص على امتداد منحنى تم وصفه بواسطة الدالة التفاضلية $y = f(x)$ كمــا مبين في الشكل 40.3.



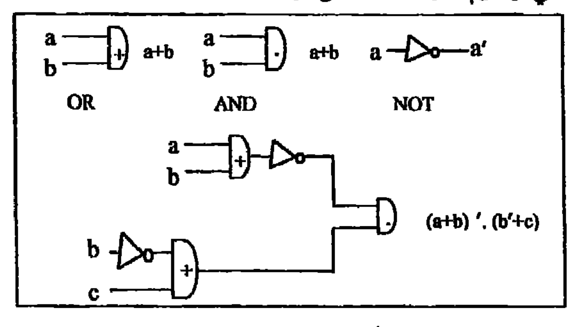
الشكل 40.3 يتبع مسار منحنى

8.3 استخدم نص أو صفات منظومة GKS، صمــم برنــامج بسيط للنشر على حاسوب منضدي (Desk Top). البرنــامج ينبغي أن يقرأ ملف إدخال لرموز وأوامر في شفرة ASCII ويقوم بإنتـــاج حـــروف طباعـــة (typeset) كمخرجات. يمكنك وضع الأوامر ضمن مدخلات ASCII وذلك باستخدام الرمز / كبادئة (Prefix) للأوامر مثال ذلك، \font (font – num) أو \v space (vert-space).

9.3 تستخدم التعابير البولينية أو المنطقية مثال ذلك

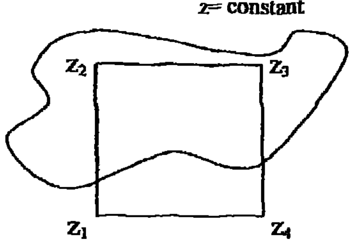
$$f=(a+b)' \cdot (b'+c)$$

استخدم ثلاثة عمليات أساسية هي : NOT ('), OR (+), AND (.) . يمكننــا عــرض أي تعبير بوليني باستخدام الرموز (أو بوابات gates) كما مبين في الشكل 41.3. أكتب برنــامج يأخذ أي تعبير منطقي ويقوم بعرضه كشكل أو مخطط.



الشكل 41.3 رموز منطقية

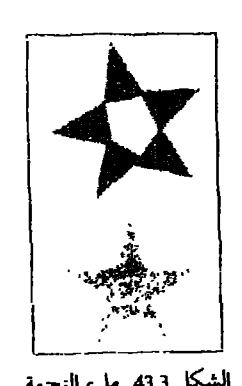
10.3 المعادلة $z = f(x,y)$ تصف سطح ما. إحدى الطرق لعرض هذا السطح هو رسم سلسلة من المنحنيات المغلقة أو الكنتورات (contours). الكنتور هو عبارة عن منحنى مغلق لقيم y,x الذي يحقق المعادلة لقيم z الثابتة. نظراً لكون هذه المعادلة ضمنية (Implicit)، تكون إحدى الطرق الممكنة لتوليد مجموعة قيم z المقابلة لكل زوج من قيم y,x على شبكة معينة متكونة من مجموعة خلايا ، كما مبين في الشكل 42.3. بالنسبة لكنتور مقابل قيمة لـ z معطاة ولتكن بين z_1,z_2 - يمكننا استكمال (Interpolate) على امتداد حافة الخلية لإيجاد نقطة على الكنتور. اكتب برنامج يقوم بتوليد الخلايا من الدالة f (x,y) ومن ثم يقوم بتوليد متعدد الخطوط لتقريب الكنتورات (المنحنيات المغلقة)، هذا التمرين صعب، ليس من الضروري أن تقوم الخلية بتحديد تقاطعات الكنتور الغامضة.



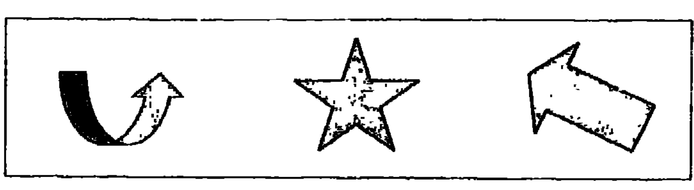
الشكل 42.3 خلايا كنتورية

11.3 نحن نقوم بتعريف داخل وخارج متعدد الخطوط من خلال تعريف خط المسح. هل يمكنك تعميم هذا التعريف بحيث لا نحتاج الإشارة إلى خطوط المسح؟ ماذا يحدث لو مر خط مسح من خلال رأس.

12.3 إن تعريفنا لمساحة الملء تنتج منها إملاء نموذج النجمة المبين في أعلى الشكل 43.3. كيف يمكننا تعريف مساحة الملء بحيث نحصل على نموذج النجمة ممتلئة كما مبين في أسفل الشكل 43.3.


الشكل 43.3 ملء النجمة

13.3 لقد تم إضافة مجموعة كيانات أولية لمساحة الملء (الشكل 44.3) إلى العديد من منظومات الرسومات. لقد تم تعريفه بواسطة قائمة عناصرها تشير إلى قوائم النقاط. وكل قائمة نقاط تقوم بتعريف مساحة ملء. هل يمكن بناء هذا الكيان الأولي من الكيان الأولى لمساحة الملء؟



الشكل 44.3 مجموعة ملء

14.3 هناك تعريف آخر لطريقة تحديد الداخل لمساحة ما وذلك باجتياز حدوده بإتجاه عقرب الساعة، وتعريف "الداخل" سيكون على يمين اتجاه مسارنا . هل تعتقد هذا التعريف عملياً بالنسبة لرسومات الحاسوب.

15.3 في كثير من الأحيان، عندما نقوم بتوليد كميات هائلة من البيانات (أحادية البعد)، يتم عرضها كسلسلة من المخططات (Cascade Plot) البيانية كما في الشكل 45.3.

يحتوي الصف الأول على أول K من نقاط البيانات ويحتوي الصف الثاني على K من نقاط البيانات التالية وهكذا. لاحظ ذلك، إن كل صف من البيانات يخفي البيانات التي خلفه. أكتب برنامجاً لتوليد مثل هذا المخطط.



الشكل 45.3 رسومات متعاقبة

الفصل الرابع

رسومات متفاعلة (Interactive Graphics)



الفصل الرابع

رسومات متفاعلة (Interactive Graphics)

مقدمة

لو ألقينا نظرة على التطبيقات الأكثر تطوراً وإثارة للرسومات الحديثة بالحاسوب ستظهر لنا معظم السمات التالية:

1- إدخال وتفاعل .

2- معالجة الصور أو مناورة مع الصور.

3- نمذجة الأشياء المنظورة المركبة والظواهر الطبيعية.

من أجل أن تكون منظومة برمجيات الرسومات نافعاً في أجواء محطة العمل الحديثة ينبغي للبرمجيات أن تمتلك قدرات تمكنها من دعم هذه المميزات.

1.4 برمجة مع تفاعل *(Programming With Interaction)*

لنأخذ على سبيل المثال تطبيق CAD متفاعل كالذي نجده في تصميم قطع الغيار الميكانيكية. في مثل هذه التطبيقات، نحن نتعامل مع نماذج لأشياء منظورة ثلاثية الأبعاد. كجزء من عملية التصميم، نقوم بإضافة وتعديل عناصر النموذج بصورة تفاعلية. أيضاً نقوم بتغيير منظور النموذج للوصول إلى أجزاء أخرى من التصميم. هذه العمليات تحتاج إلى قدرات إضافية من منظومة الرسومات.

مع أنه، على المدى البعيد، يكون من المهم معرفة ماذا ستكون رياضيات التدوير (Rotations) أو كيف تم بناء المكونات المادية للأجهزة التفاعلية، علماً بأن المبرمج قد لا يحتاج هذه المعرفة بالتفصيل من أجل كتابة برنامج. هنا، نحن لا نقصد بالتلميح إليك بأن معرفة هذه المعلومات ليست ضرورية. بالأحرى، نقترح ما يلي، يمكننا البدء بدراسة وظائف البرمجيات (Software Functionality). بصورة مستقلة عن تنفيذها.

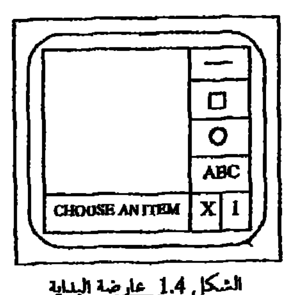
هنا الفصل سيركز على وجهين من التفاعل عالي المستوى للتطبيقات هما: التجــــزؤ والإدخال (Segmentation and Input). يتيح لنا التجزؤ بجمع كيانات أوليــة لتكويــن أشياء منظورة معروفة من قبل المستفيد. لكي تكون لنا القدرة على التعامل مـــع أشــياء منظورة بدلاً من التعامل مع كيانات أولية بصورة منفردة، تكون الخطوة الأولى هي نحــو تطوير تقنيات نمذجة متفاعلة (Interactive Modeling) ومتطورة وشاملة. بعدئذ ســنقوم بتطوير صيغ أساسية للإدخال البياني (Graphical Input) وذلك بالتعرف على الأجــهزة وإجراءات الإدخال (Input Procedures). إن إضافة قدرة التحويل ســيكون موضــوع الفصل القادم.

كما في الفصل السابق، ستتعرف هنا على البرمجيات وذلك باستخدام مثــال بســيط ولكن توضيحي: هو برنامج تخطيط الأشكال (A Shape- Layout Program). يكون هذا التطبيق من الناحية المفاهيمية مشابهة إلى معظم برامج الرسم الزيتي أو التلوين (Painting) أو برامج تخطيط الدوائر الكهربائية ومجموعة أخرى من تطبيقات متفاعلة مدارة بقائمة اختيار.

2.4 برنامج تخطيط الأشكال *(A Shape – Layout Program)*

إن اختيارنا لبرنامج تخطيط الأشكال يساعدنا في توضيح ليس فقط سمات الرسومات الإضافية لحزمة البرمجيات كالإدخال والتجزؤ، بل أيضاً يوضح لنا كيفية استخدام عــــدة وسائل تصميم قياسية في واجهات المستفيد، مثل قوائم الاختيـــار (Menus) أو رســائل للتذكير أو توجيهات (Prompting) أو شواخص (Icons) . سيكون المثال بسيطاً بقـــدر المستطاع، ويمكنك إجراء تجاربك بصورة حرة مع عدد من الاختبارات التي تم توفيرهــــا. وستعود إلى دراسة تصميم الواجهة البينية للمستفيد في نهاية الفصل، بعد دراسة آلية كتابــة برامج متفاعلة.

برنامجنا سيتيح للمستفيد القيام بتكوين صور تحتوي على أشكال قد تظهر في قائمـــة الاختيار. كذلك سيكون قادر على إضافة نصوص للصورة. أيضاً ستوفر الشاشة المعروضة إلى المستفيد توجيهات تساعده في عملية تخطيط الأشكال. كتطبيق نموذجي لهــذا المثــال البسيط قد تكون تهيئة مخططات إنسيابية (Flowcharts) للبرامج. مـــع إدخـــال بعــض



الشكل 1.4 عارضة البداية

التعديلات البسيطة، يمكن أن يصبح البرنامج مخطط لدوائر كهربائية أو رسم الصور الزيتية (الطلاء Painting).

يبين الشكل 1.4 محتوى الشاشة الابتدائية المعروضة للمستفيد. تحتوي العارضة على أربعة مساحات تمثل أربعة بوابات رؤية مميزة في برنامجنا وهي:

1- المساحة العليا لليسار: في البداية تكون هذه المساحة فارغة أو تستخدم لتوليد الصور من قبل المستفيد.

2- المساحة العليا لليمين: تظهر فيها الأشكال المسموحة في قائمة الاختيار.

3- المساحة السفلى لليسار: تظهر فيها توجيهات للمستفيد على شكل رسائل لأخبار المستفيد ماذا ستكون النشاطات المطلوبة التالية.

4- المساحة السفلى لليمين: تستخدم في التحكم أو السيطرة في حالة إنهاء البرنامج أو مسح منطقة العمل.

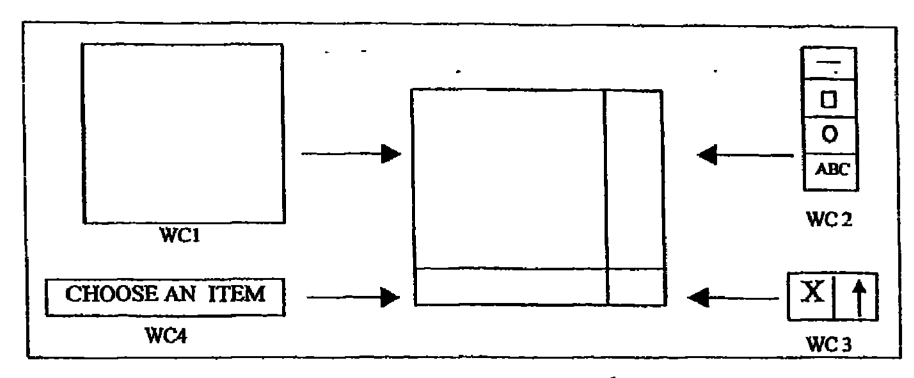
توجد هنالك عدة سمات مفقودة قد ترغب في إضافتها لاحقاً، مثال ذلك ، طريقة إبطال النشاطات أو طريقة إنتاج المخرجات على جهاز استنساخ (Hardcopy) أو طابعة (Printer).

لاحظ ذلك، في اثنين من بوابات الرؤية، نجد شواخص أو صور تمثل نشاطات أو إجراءات ممكنة. على سبيل المثال، الشاخص 'x' يسير إلى نشاط أو إجراء مسح الشاشة. قد يكون هذا الاختبار للشاخص ضعيف لأن الرمز 'x' قد لا يوصل للمستفيد معنى هذا النشاط أو كرمز يمثل مسح الشاشة. باستطاعتنا استخدام رمز آخر بدلاً من 'x' في تصميمنا مع تعديل صغير لبرنامجنا. أما رسالة البداية المعروضة تطلب من المستفيد الانتقاء من قائمة اختيار الأشكال التي ستوضع في مساحة الرسم، حالما يكتمل هذا النشاط ستظهر رسالة جديدة، في هذه المساحة.

1.2.4 اختيار نوافذ وبوبات رؤية (Choosing Windows And Viewports)

يبدأ تصميمنا بوضع تخطيط العارضة، الذي يشبه الإجراء الذي تتبعناه في برنامج رسم البيانات. العبارات التالية للبرنامج تحتوي على اختياراتنا للنوافذ وبوابات الرؤية

المتنوعة. لاحقاً، سنضع هذه الأجراءات في وحدة مستقلة (Separate Modules). سوف نستخدم أربعة تحويلات معيارية، واحدة لكل بوابة رؤية. نظراً لعدم احتواء هذا التطبيق على منظومة إحداثيات كونية (WC) مرافقة له، لذا يمكن انتقاء كل زوج من نافذة وبوابة رؤية لها نفس نسبة مربع أقصى (Aspect Ratio) واستعمال منظومة إحداثيات WC كما في الشكل 2.4.



الشكل 2.4 منظومات إحداثية كونية

```
static Glim window1 =  {0.0, 8.0, 0.0, 9.3};
static Glim viewport1  = { 0.0, 0.8,0.1,1.0};
static Glim window2 = {0.0, 2.0,0.0, 9.0};
static Glim viewport2 = {0.8, 1.0,0.1,1.0};
static Glim window3 =  {0.0, 2.0,0.0,1.0};
static Glim viewport3 = {0.8, 1.0,0.0 ,0.1};
static Glim window4 = {0.0, 8.0, 0.0, 1.0}
static Glim viewport4 = {0.0, 0.8, 0.0, 0.1};

gset – win (DRAW- TRANS, &window1);
gset – vp (DRAW – TRANS, &viewport1);

gset – win (MENU-TRANS, &window2);
gset – vp  (MENU-TRANS, &viewport2);

gset- win (CONTROL– TRANS, &window3);
gset- vp (CONTROL-TRANS, &viewport3);

gset – win (MESSAGE-TRANS, &window4);
gset –vp ( MESSAGE- TRANS, &viewport4);
```

2.24 قائمة اختيار الأشكال (The Shape Menu)

تكون خطوتنا التالية هي إعداد قائمة اختيار للأشياء المنظورة. لنفترض يتوفر لدينا كيان أولي للرسم العام الذي يقوم برسم دائرة مستخدماً صفات متعدد الخطوط الحالي. النقطتين التي تم افتراضهما ستكون مدخلاتها مركز الدائرة وأية نقطة على محيطها. نظراً لأننا سوف نقوم بإحاطة المساحات بواسطة مستطيلات، سنقوم باستخدام إجراء رسم الإطار (box-drawing). ولكن لو رغبنا برسم أطر الواحد فوق الآخر (الرسم الأخير فوق الرسم السابق) يمكننا تعريف إطار الذي يكون في الحقيقة مساحة ملء باللون الخلفي (Background Colour) يليه متعدد خطوط باللون الأمامي (Foreground Colour)، وعلى سبيل المثال:

```
fill- box (xmin, xmax, ymin, ymax)
Gfloat xmin, xmax, ymin , ymax;
{
    Gpt [5] box;
    box [0] .x = box [3] .x = box[4] .x = xmin;
    box [1] .x =box [2].x = xmax;
    box [0].y = box [1].y = box[4].y = ymin;
    box [2] .y= box [3].y = ymax;
    gset – fill- area – int – style (GSOLID);
    gset – fill- area – colr- ind (BACKGROUND);
    gfill- area (4, box );
    gset – linetype (SOLID);
     gset – line-colr – ind (FOREGROUND);
     gpolyline (5, box);
}
```

باستخدام هذا الإجراء، نستطيع الآن رسم مساحة قائمة اختيار الأشكال على العارضة. لقد تم اختيار المساحات بحيث تكون الأشكال في مراكز أطرها.

```
Gpt points [2];
Gpt loc;
static Gtext-align center = {GCENTRE-HOR, GHALF-VERT};
/*line icon */
```

```
fill- box (0.0, 2.0, 6.75,9.0);
points [0] .x = 0.5;
points [0].y= 7.875;
points [1]. x = 1.5;
points [1].y= 7.875;
gpolyline (2, points);
/ *rectangle icon */
fill- box (0.0, 2.0, 4.5, 6.75);
fill- box (0.5,1.5, 5.0625, 6.1875);

/* Circle Icon (Using Implementation Dependent GDP)*/
fill – box (0.0, 2.0,2.25,4.5);
points [0].x = 1.0;
points [0].y= 3.375;
points [1].x = 1.0;
points [1].y= 2.5;
ggdp (2,points, CIRCLE-ID, NULL)

/* Text Icon */
fill- box (0.0, 2.0, 0.0, 2.25);
gset – text – align (&center);
gset – char-ht (1.0);
loc. x = 1.0;
loc.y= 1.125;
gtext (&loc, "ABC");
```

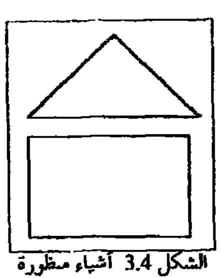
يتم استخدام عبارات برمجة مشابهة لتكوين أطر الرسائل والتحكم (Message and Control Boxes).

3.4 تعريف الأشياء المنظورة (Defining Objects)

في مثالنا، سيكون من السهل جداً للمستفيد اختيار مفردة من قائمة الاختيار إذا كان باستطاعته أن يشير إلى مساحة على الشاشة بدلاً من الإشارة إلى كيان أولي منفرد

كقطعة خط. لقد تم تهيئة هذا في رسمنا التمهيدي لقائمة الاختيار وذلك بوضع كل مفردة (المستطيل، والدائرة) فوق مساحة مملوءة بالكامل (Solid Fill Area). هكذا، إذن نستطيع اعتبار الاندماج بين مساحة الملء هذه المعرفة بواسطة الإجراء (fill – area) والكيان الأولي لتكوين الشيء المنظور بيانياً (Graphical Object). لو استطعنا تعريف مثل هذه الأشياء المنظورة ضمن منظومتنا، فقد تسهل بعض مشاكل الإدخال. بالإضافة إلى ذلك، قد نستطيع استخدام بعض الأفكار في تسهيل عمليات كإعادة رسم الشاشة وتحريك مجموعات من كيانات أولية إلى مواقع جديدة.

1.3.4 القطع (Segments)


الشكل 3.4 أشياء منظورة مقابل كيانات أولية

يعتبر الشيء المنظور بيانياً مطابق إلى ما قد نستطيع تميزه بالرؤية على العارضة كشيء منظور أو ضمن نموذجنا لتطبيق معين. على سبيل المثال، لنأخذ بنظر الاعتبار الرسم البسيط المبين في الشكل 3.4.

لو طلب منا وصف ما نراه في الشكل، لربما تكون الإجابة هكذا: "مربع ومثلث" بدلاً من "سبعة قطع خطوط". ولكن، لو نظرنا إلى الشكل عند مستوى أدنى لوصفنا الرسم بأنه عبارة عن مجموعة من قطع خطوط أو متعدد خطوط. بالنسبة للمبرمج التطبيقي، علينا تعريف هذه الكيانات بدلالة الكيانات الأولية لمنظومتنا. حالما يتم تعريف هذه ا لكيانات، نفضل العمل مع مجموعات من الكيانات الأولية.

تعتبر القطع مجموعات من الكيانات الأولية. إن اختيار أية من تلك الكيانات الأولية مرشحة أن تدخل في تركيب القطعة وتكون تحت تصرف المبرمج التطبيقي. هنالك مناظرة جيدة تساعدنا في فهم موضوع القطع وذلك باعتبار هذه القطع كأنها مماثلة إلى ملفات بسيطة– وهذا يعني، مكونات الملفات عبارة عن كيانات أولية للرسومات.

لنأخذ ملف قياسي الذي يمكن تعريفه من برنامج. لو تسألنا ماذا بالإمكان إدخاله في ملف كهذا، قد يكون الجواب تقريباً أي شيء يرغب المبرمج إدخاله في الملف. على سبيل المثال.

1- يقوم محرر (Editor) النص بتكوين ملف نص.
2- مبرمج بلغة C يستطيع تكوين ملف بيانات.
3- أيضاً يتم تخزين البرامج المهيئة للتنفيذ كملفات تحتوي على بيانات ثنائية (Binary Files).

كذلك نستطيع تكوين ملفات قد تكون محتوياتها ملفات أخرى. تتعامل البرامج مع الملفات بشكل روتيني. عندما نقوم بتشغيل (Run) برنامج مستفيد، في الحقيقة تقوم منظومة التشغيل (Operating System) بقراءة ملف يحتوي على شفرات قابلة للتنفيذ ومعالجة أي من ملفات الإدخال والإخراج المطلوبة. هكذا، من الجدير بالملاحظة أن نجد أن هنالك لا توجد ملفات تستطيع أن تحتفظ بمعلومات الرسومات وأيضاً ليس هناك لغة برمجة الرسومات تحتوي على دوال تتيح للمستفيد من تعريف ومعالجة مثل هذه الملفات. يعتبر ملف GKS الملحق، الذي سبق ذكره في الفصل الثاني هو أحد الأمثلة. القطع تمتلك كثير من خواص الملف الملحق بينما الملفات لا تمتلك ذلك.

لو بدأنا مع مفهوم ملف بلغة C الممكن استحداثه من برنامج، نجد هنالك خمس متطلبات ضرورية وهي:

1- بداية الملف.
2- نهاية الملف.
3- طريقة لتسميه أو تعريف الملف.
4- طريقة لإدخال المعلومات إلى الملف.
5- طريقة لاسترجاع المعلومات من الملف.

نفس هذه المتطلبات تنطبق على القطع. يتم تحقيق المتطلبات الثلاثة الأولى بواسطة تحديث (فتح) وإغلاق القطعة. في منظومة GKS، يتم تهيئة قطعة بواسطة الإجراء "استحداث قطعة" (create segment).

```
void gcreate – seg (seg-name)
Gint seg – name;
```

حيث يكون seg-name رقم صحيح موجب يستخدم في تميز القطعة. من الناحية العملية، يكون الثابت الرمزي عادة أكثر وضوحاً. كما في العبارات التالية:

```
#define CIRCLE 1
:
:
gcreate- seg (CIRCLE);
```

يتم إغلاق وإنهاء القطعة بواسطة الإجراء "أغلق القطعة" (close segment):

```
void gclose – seg ( );
```

بما أن قطعة واحدة فقط يمكن فتحها في كل مرة، لذا لا نحتاج وسيط أو معلمية في الإجراء gclose- seg. ما بين الفتح والغلق نستطيع تعريف كيانات أولية أو استخدام إجراءات أخرى تقوم بنفس المهمة. على سبيل المثال ، في مثالنا، سنقوم بوضع جميع الرسائل التي قد يعرضها البرنامج إلى المستفيد في قطعة، لكي نستطيع معالجتها بسهولة حسب الضرورة في برنامجنا. رسالة البداية قد تكون القطعة التالية:

```
gcreate – seg ( INITIAL-MESSAGE);
fill – box (0.0, 8.0,0.0,1.0);
gset – text – align (GCENTRE- HOR, GHALF-VERT);
gset – char – ht (1.0);
gset – text – colr- ind (FOREGROUND);
loc. x = 4.0;
loc.y = 0.5;
g text (&loc, "CHOOSE A SHAPE");
gclose – seg ( );
```

لقد تم استحداث القطعة. وباستخدام الإجراء fill – box تملأ المساحة باللون الكامل (لون الخلفية) وذلك لإخفاء أية رسومات قد تكون موجودة في هذه المساحة سابقاً. بعدئذ يتم رسم متعدد خطوط لإحاطة المساحة. أما النص، فبعد تثبيت ارتفاعه ولونه يتم مركزته في المستطيل.

علينا أن نتذكر ما يلي، عندما نقوم بتعريف كيان أولي في قطعة، تستخدم الصفات المميزة الحالية لهذا الكيان. إذن لو أهملنا تثبيت الصفات المميزة في مثالنا، تقوم المنظومة باستخدام القيم الأخيرة التي وضعت قبل استحداث القطعة.

2.3.4 القطع وانسيابية البرنامج (Segments And Program Flow)

أينما وجدت القطع أو بمعنى أدق عندما تتواجد تكون متباينة بين منظومات الرسومات. في منظومة GKS، يتم ضم القطعة إلى جميع محطات العمل الفعالة في وقت تكوين تلك القطعة. تعرض الكيانات الأولية في القطعة حالما يتم تعريفها في البرنامج. لا يمكن تعديل القطع بعد تكوينها، مع أنه توجد هناك آليات لاستحداث قطع جديدة باستخدام أجزاء من قطع موجودة. لا تتوفر هناك إمكانات لتغيير أو تنقيح قطعة موجودة. المنظومات الأخرى كمنظومة PHIGS تتخذ أسلوب مختلف. في منظومة PHIGS الهيكل (Structure– الذي يقابل القطعة في منظومة GKS) يشكل جزء في قاعدة بيانات مركزية. إن عملية عرض هذه الهياكل تسمى رصد (Posting)، وتكون مستقلة عن استحداثها. إحدى نتائج الفصل بين الاستحداث والعرض هي أنه يمكننا تنقيح الهياكل بدون الدخول في مشاكل التزامن (Concurrency) الملازمة من وجهة نظر منظومة GKS للقطع. سنقوم بالتوسع في موضوع النمذجة مع منظومتي PHIGS, GKS لاحقاً في الفصل القادم.

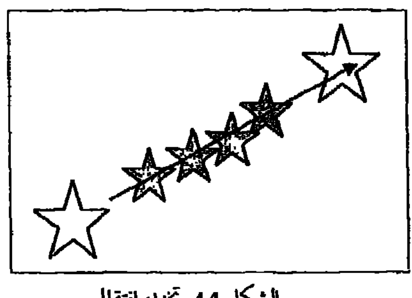
إن تدفق الكيانات الأولية من البرنامج التطبيقي إلى العارضة بحيث تكون كلاً من الكيانات الأولية المعرفة خارج القطع والمعرفة ضمن القطع تظهر على العارضة. الفرق الرئيسي في تناول هذه الكيانات الأولية يظهر عندما تقوم فعالية معينة بتغيير الشاشة (مثال ذلك، تحديث الشاشة) أو عندما نقوم بتغطية شيء ما موجود على العارضة مسبقاً. يتم إرسال الكيانات الأولية الموجودة خارج القطعة (بعد إجراء عملية التقليم الضرورية) مرة واحدة بالضبط. هكذا إذن، لو تم تغيير كيان أولي معروض على الشاشة، كما يحدث هذا في حالة وضع مساحة ملء كلية فوقه، حيث لا يمكن استعادته، إلا بإعادة توليده من برنامج المستفيد. بالنسبة للقطع، نظراً لأنها مخزونة، يمكن إعادة عرضها، لربما في مواضع جديدة مع صفات مميزة مختلفة. أما من الجانب غير الإيجابي في استخدام القطع، هناك كمية محسوسة من العمل الإضافي (Overhead) تحتاجها عملية التجزؤ (Segmentations). دائماً البرنامج الذي لم يستخدم قطع سينفذ أسرع تقريباً من مثيله الذي يستخدم القطع. قد يتلاشى هذا الفرق إذا توفر إسناد مباشر للتجزؤ من قبل المكونات المادية.

الآن لننتقل إلى كيفية استطاعتنا استخدام القطع. لا يمكننا عمل الكثير مع إجرائـــي الاستحداث والإغلاق لوحدهما. لنعود إلى مبدأ المناظرة مع الملفات، حيـــث نرغـــب أن تكون لدينا القدرة على حذف القطع التي لا نحتاجها بعد، وكذلك إعادة تسمية القطـــع. يمكن تحقيق هاتين العمليتين من خلال الإجراءين التاليين:

```
void gdel-seg (seg-name)
void grename- seg (old – seg-name, new – seg- name)
```

3.3.4 تخزين انتقالي (مرحلي) Buffering

يعتبر الحذف وإعادة التسمية من العمليات الرئيسية في آلية التخزين المرحلي الـــذي تتيح للمستفيد التحكم بالأشياء المنظورة على العارضة. لنفترض لدينا شيء منظور معـرف في قطعة مسماة S. واستجابة إلى مدخلات المستفيد، ليكن من خلال جـــهاز الإدخـــال الفأر (Mouse)، حيث نرغب بنقـــل الشـــيء المنظور STAR إلى موقع جديد كما هو مبـــين في الشكل 4.4.



الشكل 4.4 تخزين انتقالي

يمكننا تعريف قطعة مسماة NEWSTAR في الموقع الجديد. سنقوم بحذف القطعة القديمـــة STAR، ولو افترضنا أنه تم الإعـــداد بصـــورة صحيحة للتحكـــم في الشاشـــة والمعلميـــات الأخرى، لربما نتوقع للقطعة الجديدة أن تظهر على الشاشة فقط. مع ذلك، هذه الطريقـــة تخلق مشكلة وهي، نظراً لما تبقى من البرنامج، علينا أن نتذكر الاسم الجديد للقطعـــة. إن عملية إعادة التسمية تحل هذه المشكلة. برنامج التخزين المرحلي له الهيكلية التالية:

```
gcreate – seg (newstar);
:                 /* create new segment elements */
gclose- seg (NEWSTAR);
gdel – seg (STAR);
grename – seg (NEWSTAR, STAR);
```




لا يخبرنا هذا المثال متى ستظهر التغييرات علــى العارضــة بــالضبط. في منظومــة رسومات التي تكون قادرة على إعادة رسم جميع القطع في دورة إنعاش واحدة (نموذجيـا تستغرق الدورة سدس الثانية)، في الواقع يتم تنفيذ سلسلة القطع آنياً بــدون الحاجــة إلى تفاعل المستفيد. إن هذه الاستجابة الذاتية تعرف بـ "تجديد الشاشة الضمني " (Implicit Screen Regeneration) ويمكن إعدادها كجزء من إجراء التمهيد. على منظومة بطيئــة، قد يكون التجديد الضمني ضاراً، لأن كل تغيير قد يتطلب إعادة رسم العارضة بصــورة كاملة. في مثل هذه الحالة، يفضل تجديد الشاشة المحدد أو الصريــح (Explicit Screen Regeneration)، الذي يتطلب من المستفيد أن يأخذ إجراء محدد قبــل إعــادة رســم الشاشة. قد يتيح إجراء كـ "إعادة رسم جميع القطع على محطة العمــل " (redraw all segments on workstation) لبرنامج التطبيق من إعادة رسم العارضة عند الضرورة.

4.4 الصفات المميزة للقطع *(Segment Attributes)*

نظراً لكون القطع هي تعريف الأشياء المنظورة بيانياً، لذا تكون صفاتها المميزة هــي خواص الشيء المنظور بأكمله. تتضمن هذه الخواص ما يلي:-

1- الرؤية (المنظورية) (Visibility)
2- الأسبقية (Priority)
3- الاكتشافية (قابلية الاكتشاف) (Detectability)
4- إظهار الأجزاء ذات الأهمية (جزء من الصورة الأشد إضاءة). (Highlighting)
5- التحويل (Transformation).

بخلاف عديد من الصفات المميزة للكيانات الأولية، التي تكون ملازمة للكيـان الأولي منذ وقت تكوينه، تكون الصفات المميزة للقطعة قابلة للتغيير أثناء تنفيذ البرنامج.

1.4.4 الرؤيــة (Visibility)

من السمات الرئيسية للقطع هي إمكانية إزالتها من العارضة وإعادة عرضها لاحقــاً. في تطبيق معقد، قد نرغب في تعريف جميع القطع من البداية، حتى لو كانت الحاجة إليــها غير ضرورية عند وقت تعريفها . تلك كانت خطتنا بالنسبة للرسائل في برنامجنا تخطيــط

الأشكال (Layout Program). أصبحت هذه الخطة ممكنة بفضل الإجراء "ثبت الرؤيـــة" (gset-vis) وذلك بالسماح في استحداث قطع تكون في البداية غير مرئية. عـــادة يتـــم تنشيط هذا الإجراء مباشرة بعد استحداث القطعة وقبل تعريف أياً من الكيانات الأولية.

لنأخذ بنظر الاعتبار جزء من البرنامج التالي، الذي يقوم بتعريف الرســـالة الثانيـــة لبرنامجنا تخطيط الأشكال. تقوم الرسالة بالطلب من المستفيد أن يدخل نقطة النهاية لقطعـــة الخط، وسوف تعرض هذه الرسالة في حالة اختيار شاخص الخط (Line Icon) في قائمـــة الاختيارات.

```
gcreate- seg (FIRST – END);
gset – vis (FIRST – END, GINVIS);
fill – box (0.0, 8.0, 0.0, 1.0);
gset – text – align {GCENTRE-HOR, GHALF-VERT};
gset – char-ht (1.0);
gset – text- colr- ind (FOREGROUND);
loc.x = 4.0;
loc.y = 0.5;
g text (&loc, "CHOOSE FIRST ENDPOINT OF LINE");
gCLOSE – SEG (  );
```

لو افترضنا أن المنظومة تم تشكيلها بطريقة تقوم باستحداث نفسها بأسرع ما يمكـــن في حالة تحويل قطعة غير مرئية إلى قطعة مرئية وحالاً يتم رسم القطعة علـــى الشاشـــة. تعطينا هذه التقنية طريقة بسيطة للسيطرة على عرض الرسائل في مثالنا هذا. فضـــلا مـــن تفحص الكيانات الأولية للرسائل بصورة متكررة استجابة إلى مدخلات المستفيد، نقـــوم بدلاً من هذا بتعريف كل الرسائل الضرورية في قطع غير مرئية في البداية. وبعدئـــذ يتـــم تغييرها إلى قطع مرئية حسب الحاجة.

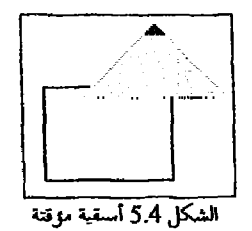
2.4.4 الأسبقية (PRIORITY)

يتم عرض الكيانات الأولية التي تكون خارج القطع مرة واحدة فقط، والمبرمج يمكنـه التأكد من ذلك تماماً من الترتيب الذي ستظهر فيه الكيانات. أما بالنسبة للكيانات الأوليـة في داخل القطع وخاصة عندما يتم تجديد الشاشة ضمنياً حيث تكون الحالة أقل وضوحـــاً بسبب تفاصيل عملية التنفيذ كطريقة التخزين المرحلي الداخلية لكل محطة عمل حقيقيـــة. قد تسبب قلة الوضوح هذه مشكلة خطيرة. على سبيل المثال، كمـــا رأينـــا في الفصـــل

الثالث، ظهور تداخل كيانات أولية لمساحات الملء والتي تكون معتمدة على الترتيب الذي يتم فيه رسم هذه الكيانات.

توجد هنالك عدة طرق ممكنة للتحكم بترتيب عرض أي من القطع، لقد استخدمت جميع هذه الطرق في منظومات الرسومات المختلفة. تكون أحد هذه الطرق البسيطة هو أن ندع اسم القطعة الذي يكون رقماً صحيحاً يحدد هذا الترتيب. طريقة أخرى قد تكون استخدام الترتيب الأصلي للتكوين. يمكننا ترك الترتيب غير محدد وندع المكونات المادية باتخاذ القرار. لكل طريقة لها فوائدها ومساوئها. منظومة GKS تتيح للمستفيد أن يحدد الترتيب، أينما يكون ضرورياً، وذلك بتثبيت الأسبقية لأي قطعة من خلال الإجراء "ثبت أسبقية القطعة " (set segment priority):

```
void gset – seg- pri (seg-name, seg – pri)
```


الشكل 5.4 أسبقية مؤقتة

حيث يكون متغير الأسبقية من نوع Gfloat وتقع بين0.0 و 1.0 يكون أعلى أسبقية 1.0. على سبيل المثال، المثلث والمربع في الشكل 5.4، لو أردنا إظهار المثلث فوق المربع، لربما يكون شكل البرنامج كما يلي:

```
gcreate – seg (TRIANGLE );
triangle (  ); /* defines a filled triangle */
gclose – seg (  );
gcreate – seg (BOX);
box (  ); /* defines a filled box */
gclose – seg (  );
      :
gset – seg – pri (TRIANGLE, FRONT);
gset – seg – pri (BOX, BACK);
gredraw- all – seg- ws (WS);
```

هنالك استخدام آخر لأسبقيات القطع يحدث عند الإدخال. لنفترض نحن قمنا باستخدام جهاز إدخال يدعى التقاط (Pick – سنقوم بشرحه في البند القادم) الذي يحدد لنا موقع القطعة. ماذا سيحدث لو (كما مبين في الشكل 5.4) كانت لدينا قطعتين

متداخلة وتم وضع جهاز الالتقاط على المساحة المتداخلة ؟ يكون من المحتمل انتقاء أحد القطعتين بواسطة جهاز الالتقاط. كما ند نتوقع ، أنه يمكننا حل هذا التضارب في عدة طرق أو قد يترك القرار إلى التنفيذ، تقوم المنظومة GKS بحل هذه المشكلة وذلك باستخدام أسبقية القطع، وإعادة رمز تعريف القطعة ذات الأسبقية العليا في المنطقة المتداخلة.

3.4.4 صفات مميزة أخرى (Other Attributes)

1- إن صيغة إظهار الأجزاء ذات الأهمية (Highlighting) تسمح بعرض قطعة ما بطريقة غير معتادة (Abnormal)، وتستخدم هذه الصفة عموماً لجلب انتباه المشاهد في برنامجنا تخطيط الأشكال، يمكننا استخدام هذه الصفة لتوجيه انتباه المستفيد إلى قائمة الاختيار التي تكون مناسبة في وقت معين.

إن صفة إظهار الأجزاء ذات الأهمية عبارة عن صفة ثنائية (Binary Attribute) التي يتم تثبيتها بواسطة الإجراء "ثبت صفة الإظهار للقطعة (set segment highlighting). يتضمن طريقة صفة الإظهار هذه مثلاً بجعل القطعة أن ترمش (Blink) أو عرضها بشدة إضاءة إضافية. من الواضح أن طريقة الإظهار هذه تعتمد على إمكانيات محطة العمل الحقيقية وقرار المنفذ في كيفية استخدام هذه الصفة.

2- أيضاً صفة الاكتشافية أو قابلية الاكتشاف (Detectability) تستخدم في الإدخال. إذا كانت قطعة غير قابلة للاكتشاف، لا يمكن اختيارها بواسطة جهاز الالتقاط. تكون هذه الصفة نافعة بصورة خاصة، كما بدى واضحاً في برنامجنا تخطيط الأشكال. في البداية، نرغب من المستفيد أن يقوم بانتقاء أحد الشواخص من قائمة الاختيار وأن لا يقوم باختيار قطعة الرسالة. إذن، لو أردنا جعل قطعة الرسالة غير قابلة للاكتشاف، سيتطلب من منظومة الرسومات أن لا تقبل هذا الاختيار حتى لو حاول المستفيد خطأً أن يختاره.

3- كذلك يمكن إعادة توجيه القطع أو تحريكها حول الشاشة بواسطة التحويلات.

هكذا يمكننا تعريف قطعة في موقع مناسب وبأي حجم واتجاه مطلوبين. وبعدئذ يمكننا استخدام تحويل على القطعة لعرضها بحجم واتجاه وموقع آخر. سوف نصف هذا

التحويل رياضياً بواسطة مصفوفات التي يمكن اعتبارها صنفة للقطعة. ومن أجل بحث موضوع التحويلات يتوجب علينا تطوير الرياضيات الضرورية لها. سنؤجل هذا البحث إلى الفصل الخامس.

5.4 الإدخال (Input)

لنعود إلى برابجنا تخطيط الأشكال، الآن نستطيع التوجه إلى دراسة مشاكل التفاعل مع المستفيد. توجد هنالك على الأقل ثلاثة حالات مختلفة في برنابجنا تحتاج إلى مدخلات وهي:

1- عندما يقوم المستفيد بالاستجابة إلى رسالة البداية لاختيار مفرده من قائمة اختيار الأشكال، وذلك بالإشارة إلى شاخص في تلك القائمة. يكون النوع المألوف من المدخلات المعادة إلى البرنامج نتيجة هذا الاختيار هو رمز تعريف شيء منظور أو قطعة.

2- لنفترض أن المستفيد اختار شاخص الخط. ستظهر رسالة تطلب منه أن يدخل أحد نقطتي النهاية لقطعة الخط المطلوبة. هنا تكون المدخلات المطلوبة عبارة عن إحداثيات موقع.

3- لو كان مستفيدنا قد اختار شاخص النص، سيوجه بإدخال صف من الرموز وهذا هو الشكل الثالث من المدخلات.

لقد أدركت منظومات الرسومات الحاجة إلى أنواع مختلفة من المدخلات. بدلاً من أن يقوم المستفيد بكتابة برنامج لإدخال موقع وإيجاد قطعة يكون الموقع جزء منها. توفر المنظومات الحديثة إجراءات تقوم بتزويد الصيغ المطلوبة للمدخلات بصورة مباشرة. هذه الإمكانات ليست فقط تعمل على تبسيط مشكلة البرمجيات، ولكن أيضاً تتيح باستخدام مكونات مادية خاصة أينما تكون متيسرة.

1.5.4 إدخال منطقي مقابل حقيقي (Logical Versus Physical Input)

بالضبط كما يعمل المبرمج مع عارضات منطقية بدلاً من حقيقية، يكون الإدخال أيضاً منطقيا. يتم تعريف أجهزة الإدخال المنطقية بواسطة نوع المدخلات أو قياس (Measure) الذي يتم إعادته إلى البرنامج.

تشمل أجهزة الإدخال الحقيقة الفأر والقلم الضوئي ولوحة البيانات ولوحة مفاتيح لمحطة طرفية. يستطيع كل جهاز أن يوفر نوع واحد أو أكثر من الفئات المنطقية (Logical Classes) للمدخلات. على سبيل المثال، بالإمكان استخدام قلم ضوئي للإشارة إلى شاخص في قائمة اختياراتنا، ونستخدم لاحقاً للإشارة إلى موقع على الشاشة. باستطاعة جهاز الفأر أن يقوم بنفس المهمات. كلا هذين الجهازين للإدخال لا يمكنها بسهولة تجهيز برنامج تطبيقي بصفوف من الرموز (Strings Of Characters). لكن من الناحية الأخرى، من الممكن استخدام لوحة مفاتيح قياسية مع مفاتيح سهمية (Arrow Keys) بالقيام بجميع هذه المهمات الثلاث. كل واحد من أجهزة الإدخال الحقيقية الثلاث هذه تمتلك قدرات مختلفة. من وجهة نظر المبرمج نفضل عدم القلق حول هذه القضايا. نحن نرغب في كتابه برنامج حيث لا يحتاج إلى تغييره في حالة تبدل الأجهزة الحقيقية، لذا استخدام الأجهزة المنطقية عند مستوى المبرمج يؤدي إلى تجنب هذه المشاكل. إذن عند كتابة برنامج تطبيقي، فقط نحتاج معرفة هل أن المنظومة تسند جميع الأنواع أو الفئات المنطقية الضرورية.

2.5.4 الفئات المنطقية للإدخال (Logical Input Classes)

منظومة GKS وعديد من المنظومات الأخرى تقوم بتصنيف الإدخال إلى ستة أنواع أو صفوف منطقية:

1. الالتقاط (Pick).
2. محدد موقع (Locator).
3. صف من الرموز (Strings).
4. اختيار (Choice).
5. ضربة أو شوط (Stroke).
6. مخمن أو مقدر (Valuator).

الأنواع الثلاثة الأولى هي تلك التي نحتاجها في نموذجنا للمشكلة.

ما يلي تعريف أولي لهذه الأنواع أو الأصناف:

1- يقوم جهاز الالتقاط (Pick – أو التأشير) بإعادة رمز تعريف القطعة التي تم الإشارة إليها بواسطة جهاز إدخال حقيقي، كالفأر أو القلم الضوئي. بصورة عامة، التأشير بالقرب من أي جزء لأي كان أولي في القطعة سيعيد رمز تعريف القطعة للبرنامج.

2- يزود محدد الموقع (Locator) بزوج من قيم y,x في فضاء WC ثنائي الأبعاد كمنظومة GKS. بالتأكيد يكون محدد موقع ثلاثي الأبعاد هو امتداد طبيعي إلى منظومة الرسومات ثلاثية الأبعاد. إن التنفيذ الحقيقي لمحددات المواقع عادة تستخدم بنفس الجهاز للالتقاط، الذي سوف نرى فائدته المميزة في عدد من البرامج.

3- مدخل صف من الرموز (String Input) يقوم بتجهيز صف من الرموز. يكون عادة التنفيذ الفعلي لهذا النوع من الإدخال هو استخدام لوحة مفاتيح (Keyboard)، مع أن بالإمكان استخدام ملفات قرصية (Dise Files) للتزود بالرموز. بالحقيقة يمكن وضع لوحة مفاتيح مرسومة بيانياً على العارضة والتي يتم تشغيلها من قبل المستفيد بواسطة جهاز الفأر.

4- يقوم جهاز الاختيار (Choice Device) بانتقاء اختيار واحد من عدد من الاختيارات الممكنة ويعتبر امتداد منطقي للأزرار (Buttons). يمكن أن يتراوح التنفيذ الفعلي لهذا الجهاز من مفاتيح على لوحة المفاتيح إلى أزرار مرسومة بيانياً التي يتم تشغيلها بواسطة الفأر أو القلم الضوئي.

5- يستخدم الشوط أو الضربة (Stroke) للحصول على مجموعة متسلسلة من المواقع، كالذي قد نحتاجه في تطبيق التلوين (Painting Application). بطريقة أو بأخرى، يعمل الشوط كمحدد موقع في دورة (Loop). إن اعتبار الشوط كنوع منطقي مستقل يتيح لنا تنفيذ دالة الشوط على جهاز حقيقي مستقل وتنفيذ الدورة بواسطة المكونات المادية بدلاً من البرمجيات.

6- وأخيراً يتيح المقدر أو المخمن (Valuator) إدخال أرقام حقيقية منفردة. أنه يعمل كقرص مدرج أو مزولة (Dial). إن التنفيذ الفعلي لهذا الجهاز قد يتراوح من ربط قرص مدرج بحاسبتنا من خلال محول تناظري -إلى - رقمي (ADC) إلى طريقة إدخال بسيطة من لوحة المفاتيح.

3.5.4 قياس وقدح (Measure and Trigger)

إن العلاقة بين الأجهزة المنطقية والحقيقية للإخراج تكون واضحة تماماً وأحادية الاتجاه (One-Direction) يتم تعريف الإخراج منطقياً ويمرر من خلال سلسلة تحويلات لتظهر كمخرجات حقيقية على جهاز عرض حقيقي. باستثناء بعض الأشياء الدقيقة في التوقيت، كتحديد متى تم تحديث العارضة بالضبط، هذه العملية هي بالحقيقة سهلة التصور.

أما بالنسبة للإدخال فالعملية تكون نوعاً ما أكثر بقليل للأسباب التالية:

1- تغطية أنواع متعددة من المدخلات.

2- أنماط تشغيل مختلفة.

3- النقل ذات الاتجاهين (Two-Dimensional Transfer) بين البرنامج التطبيقي وأجهزة الإدخال الحقيقية.

4- يمكننا نمذجة هذا التفاعل من خلال عمليتين:

- عملية قياس (Measure Process).
- عملية قدح (Trigger Process)

يعرف جهاز الإدخال المنطقي بواسطة قياس وقدح وتوجيه أو الصدى (prompt /echo).

ما يلي هو تعريف للقياس والقدح:

1. يعتبر القياس لجهاز هو عبارة عن مجموعة قيم منطقية معادة من عملية القياس منفذة على الجهاز الحقيقي. إذن، القياس لجهاز التقاط سيتضمن رمز تعريف قطعة، بينما القياس لتحديد موقع يشمل إحداثيات في فضاء WC. في كلتا الحالتين يعاد قياس الحالة (Status Measure) أيضاً.

سيقوم إجراء الإدخال بتمهيد عملية القياس. وكجزء من عملية التمهيد، قد تظهر على العارضة مزلقة (Cursor) كإشارة للبدء أو الحث (Prompt)، ولربما يتم تثبيت القيمة الابتدائية للقياس. وقد تظهر (صدى) القيمة الحالية للقياس على العارضة. يتم الخروج من عملية القياس عند إنهاءه (Terminated) أما بواسطة عملية القدح أو من خلال برنامج المستفيد.

2. يعتبر قدح الجهاز المنطقي هو عبارة عن جهاز حقيقي له القدرة علـــى توليـــد إشارات (إحداث قدح- Trigger Events). المولدات النموذجيـــة لإطـــلاق القدح (Trigger Firings) تشمل زر موجود على جهاز الإدخـــال كالفـــأر ومفتاح الإدخال الموجود على لوحة المفاتيح. تعتبر إشارة القـــدح مشـــابه إلى إشارة المقاطعة (Interrupt)، وهو عبارة عن حدث يدل عن زمـــن. يمكننـــا استخدام الإشارة لإنهاء إجراء الإدخال واعتبار الحالة الجارية (Current State) لعملية القياسي هي المدخلات المطلوبة، أو يمكن استخدامه لوضع القياس الحالي في طابور (Queue). سيعتمد الاستخدام الدقيق لعمليتين القدح والقياس علـــى نمط الإدخال.

4.5.4 أنماط الإدخال (Input Modes)

عندما نكتب برنامج بسيط بلغة C الذي يحتاج إلى مدخلات، مثلاً مـــن خـــلال الإجراء "scanf":

```
scanf ( "%d", number);
```

حيث نقوم باتخاذ بعض القرارات حول كلاً من المصدر وتوقيت الإدخال. يكـــون الجهاز عبارة عن جهاز إدخال قياسي كالمحطة الطرفية التي نستخدمها. في الحقيقـــة، قـــد يكون الجهاز القياسي هو أياً من المحطات الطرفية الحقيقية المتنوعة أو ملف قرصـــي (Disc File) أو حتى مخرجات برنامج آخر، وهذا يعتبر مثال آخر نستخدم فيه جـــهاز إدخـــال منطقي. قد يصل البرنامج إلى حالة التوقف في حين ينتظر اســـتقبال مدخـــلات أخـــرى ضرورية. إذا كانت المدخلات آتية من محطة طرفية، يقوم المستفيد بطبع الرموز وينـــهي الإدخال وذلك بالضرب على مفتاح الإدخال أو العودة (Enter Or Return)، في هـــذا المثال ، تقوم عملية القياس بإظهار الرموز على العارضة. أما بالنسبة لإشارة القـــدح (أي الضرب على مفتاح الإدخال) تكون مطلوبة لإشعار البرنامج بنهايـــة الإدخـــال وعلـــى البرنامج أن يواصل التنفيذ.

توجد هنالك طرق أخرى للبرنامج كي يحصل على مدخلات رمـــوز (Character Input). في كثير من الأحيان، يطلب البرنامج من المستفيد المصادقة على اختيار بواســـطة طبع رسالة مثل "اضرب على المفتاح Y لتستمر" وحالما يتم لمس هـــذا المفتـــاح يســـتمر

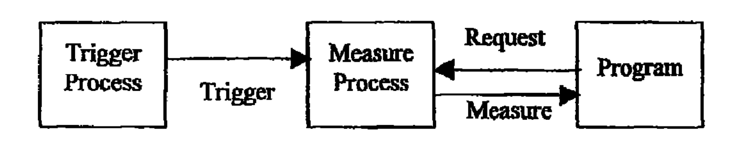
البرنامج بالتنفيذ. في مثل هذه الحالة لا نحتاج إلى الضرب على مفتاح الإدخال (Enter Key). هذان المثالات يوضحان نمطين مختلفين من الإدخال.

في مجال الرسومات بالحاسوب هنالك ثلاثة أنماط مهمة للإدخال:

1- طلب (Request)
2- عينة (Sample)
3- حدث (Event)

يتم اختيار النمط من خلال الإجراء " ثبت نمط الإدخال " (set input mode)، وفيما يلي هو تعريف لهذه الأنماط:

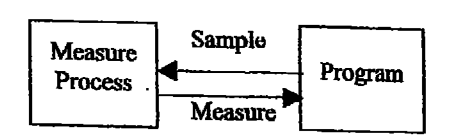
1- يعتبر نمط الطلب (Request Mode) أبسط الأنواع، وسنقوم باستخدامه في برنامجنا تخطيط الأشكال (Layout Program). عند تنفيذ إجراء طلب (Request Procedure)، كطلب تحديد موقع (Request Locator)، تبدأ عملية قياس على الجهاز المخصص. تقوم إشارة القدح بإنهاء هذه العملية وتعاد الحالة الجارية (Current State) لعملية القياس إلى البرنامج. عند هذه المرحلة، يستطيع برنامج المستفيد أن يواصل التنفيذ مع القيم المعادة إليه. الشكل 6.4.



2- الشكل 6.4 الإدخال في نمط الطلب

2- يبين فيه تسلسل العملية. إن فوائد نمط الطلب للإدخال هو إمكانية تحديد وضع الجهاز أو إدخال البيانات في أي وقت قبل قدح الجهاز . لذا يستطيع المستفيد من تصحيح ضربات المفاتيح أو تحديد موقع الجهاز بعناية حسب الرغبة.

3- في نمط العينة (Sample Mode)، تأخذ المدخلات من الجهاز حالما يتم تنفيذ إجراء الإدخال. تأخذ عملية العينة (Sample Process) القياس الجاري بدون استخدام إشارة القدح كما مبين في الشكل 7.4.

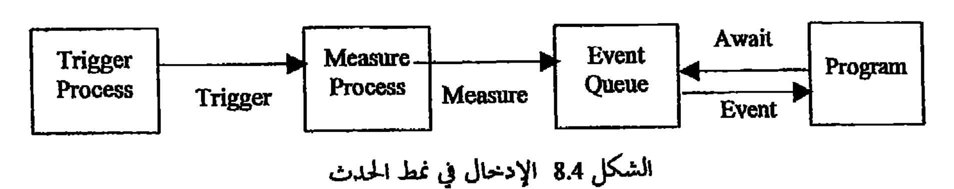


الشكل 7.4 الإدخال في نمط العينة

إذن، البرنامج سيرى المدخلات من آخر فعالية قبل تنفيذ إجراء نمط العينة للإدخال (Sample-Mode Input) بالنسبة لجهاز تحديد الموقع ينبغي تحديد موضع جهاز الإدخال الحقيقي قبل إصدار الأمر. وأما بالنسبة للوحة المفاتيح، تقوم عملية القياس بإعادة حالة مخزنة الانتقالي. في نمط العينة، العناية بالتمهيد (Initialization) تكون مسألة حاسمة، لأنها لا تشبه نمط الطلب وذلك ليس بإمكان المستفيد التحكم أو تصحيح فعالياته بدقة.

4- في كلا النمطين الطلب والعينة، قمنا بطلب الإدخال من جهاز معين. عادة تمتلك منظومات الرسومات الحديثة أجهزة إدخال متنوعة. لتأخذ مثال محاكي الطيران. يرى الطيار صور تم إسقاطها على الشاشات. وتكون في متناول يده أجهزة إدخال كعصا التحكم (Joystick) وأقراص مدرجة أو مزولات (Dials) متنوعة، ومفاتيح (Switches). وفي أي وقت، البرنامج الذي يسيطر على المحاكي ينبغي أن يكون قادراً على الاستجابة لمدخلات أياً من هذه الأجهزة، نظراً لكون الطيار يمكنه أن يشرع بأي عدد من الفعاليات أو النشاطات في أي وقت. إذن كلا نمطي الإدخال العينة والطلب لا يناسبان هذه الحالة. هكذا النمط الثالث للإدخال، نمط الحدث يمكن أن يستخدم لمثل هذه الحالة (مدخلات الحدث- event input).

في نمط الحدث للإدخال (Event-Mode Input)، كما مبين في الشكل 8.4.



الشكل 8.4 الإدخال في نمط الحدث

كل مرة يحدث قدح، توضع الحالة الجارية (Current State) لعملية القياس في طابور الأحداث (Event Queue)، ومعه المعلومات الضرورية لتشخيص نوع القياس والجهاز المنطقي الذي قام بتوليده. حالما تبدأ عملية القياس، تأخذ بالاستمرار حتى يتم إيقافها من قبل البرنامج. وبإعطاء إشارات قدح أخرى تضيف ببساطة إحداث أحداث أخرى للطابور. يتم فحص الأحداث التي وضعت في الطابور من قبل برنامج التطبيق، بصورة مستقلة عن عملية تسجيل الأحداث (Logging).

كما قد نتوقع، نمط الحدث هو أكثر صعوبة بالتنفيذ . حيث ينبغي إدامة وفحص الطوابير لأي حدث حصل، إذا كان فعلاً هذا الحدث قد حصل. سوف نقوم بدراسة نمط الحدث للإدخال في نهاية هذا الفصل.

5.5.4 التغذية المرتدة للتوجيه (الحث، الصدى والحالة) (Prompt, Echo, and Status Feedback):

إن استخدام الإدخال بصورة فعالة ينبغي تزويد المستفيد بالتغذية المرتدة (Feedback). عندما يقوم المستفيد بالإشارة إلى موقع بواسطة وضع الفأر أو تحريك إبرة تسجيل (Stylus) على لوحة البيانات، عادة يرى المستفيد موقع الجهاز مشار إليه بواسطة منزلقة (Cursor) على العارضة. لذلك بدون التغذية المرتدة، تكون عملية تحديد موقع بدقة صعبة التحقيق. عندما نقوم بطبع صف من الرموز على لوحة المفاتيح، نحن في الواقع نرى دائماً عرض صف من الرموز، كما تم طبعها. إذا كان لديك شك في أهمية آلية التغذية المرتدة هذه، ما عليك إلا محاولة طبع هذه الجملة بدون النظر إلى الشاشة أو لوحة المفاتيح.

هنالك ثلاثة طرق للحصول على تغذية مرتدة حيث لها أهمية خاصة وهي:

١. رسائل حث أو توجيهات (Prompts)
٢. الأصداء (Echoes)
٣. الحالة المعادة (Status Return)

وفي ما يلي تعريف لهذه الطرق:

1- قد تكون رسالة الحث أو التوجيه (Prompt) عبارة عن ظهور المنزلقة (أو أي علامة أخرى) على الشاشة أو خط مرسوم من نقطة ثابتة إلى الموقع المنطقي لجهاز الإدخال كما هو مخطط على الشاشة.

2- استخدام الصدى (Echo) قد يتراوح بين استنساخ الرموز على الشاشة أثناء إدخالها إلى توهج أو إيماض (Flashing) القطعة بعد اختيارها بواسطة جهاز الالتقاط. يمكننا اعتبار التوجيهات والأصداء كصفات مميزة للإدخال. تمتاز هاتين الصفتين بالاعتمادية على التنفيذ ، لأن الصفات المادية للجهاز هي التي تحدد ما نوع التوجيهات والأصداء التي يمكن الحصول عليها.

3- إن إعادة متغير الحالة (Status Variable) استجابةً إلى أي محاولة لاستخدام الإدخال يعطي صيغة مختلفة للتغذية المرتدة. لنفترض قام المستفيد بقدح جهاز التقاط وذلك بدفع الزر الموجود على الجهاز الحقيقي ولكنه لم يشير نحو قطعة معينة. قد تسبب هذه الحالة بعض الصعوبات، لأن المستفيد (بعد قدح جهازه للإدخال) قد لا يمكنه التمييز فيما إذا أخطأ في التأشير نحو الشيء المطلوب أو هناك مشكلة في برنامجه. إذن، ينبغي على المنظومة جيدة التصميم أن تتيح للمبرمج كتابة برامج تطبيقية تمكنه من اكتشاف هذه الأنواع من الحالات واتخاذ الإجراءات المناسبة. في هذا المثال، يمكن عرض رسالة تقول "حاول مرة ثانية" . هنا تكون الحالة المعادة للمتغير GNONE مقابل GOK، يتيح لبرنامج التطبيق في اكتشاف أن الجهاز تم قدحه ولكنه لم يشير نحو أي قطعة. سنقوم باستخدام عودة الحالة بصورة واسعة للتحكم في انسيابية البرامج.

6.5.4 برمجة الإدخال (Programming Input)

في بعض النواحي، يكون الإدخال مفاهيمياً مشابه إلى الإخراج، إلا أن المعالجة تكون معكوسة. يقوم جهاز الإدخال الحقيقي بإنتاج مدخلات في منظومة إحداثيات DC. بالنسبة لمدخلات أجهزة تحديد الموقع والضربة (Locator and Stroke)، يتم تحويل المعلومات من DC إلى NDC ومن ثم إلى WC. وأخيراً يستلم المبرمج المدخلات في منظومة الإحداثيات المستخدمة في البرنامج التطبيقي.

بالنسبة لمبرمج التطبيقات تأتي المدخلات من محطات عمل منطقية ، التي ينبغي أن تفتح وتنشط وتوقف ومن ثم تغلق. بالإمكان إجراء هذه العمليات من ضمن إجراءاتنا init و finish. مع ذلك، هنالك بعض المشاكل الإضافية علينا الاهتمام بها والتي تتعلق بمحطات العمل للإدخال أو للإدخال والإخراج معاً. هذه المشاكل قد تكون:

1- بخلاف محطة العمل للإخراج التي تحتوي على عارضة واحدة، حيث تحتوي محطة العمل للإدخال أو للإدخال والإخراج معاً أكثر من جهاز إدخال واحد لنوع منطقي معين. إذن ينبغي على دوال الطلب أو العينة للإدخال الإشارة ليس فقط إلى أي محطة عمل بل أيضاً إلى أي جهاز على محطة العمل سوف يتم استخدامه.

2- إن توفر رسائل الحث والأصداء غالباً ما تجعل أجهزة الإدخال مرغوباً فيها لإعادة تمهيدها لغرض تغير موقع رسالة الحث أو نوع الصدى، علماً أن أجهزة الإدخال تكون في متناول البرنامج عندما تكون محطات العمل مفتوحة وفعالة.

سوف نقوم بدراسة هذه القضايا عندما نعود إلى برنامجنا تخطيط الأشكال. أولاً، دعنا نتعرف على أجهزة الإدخال الحقيقية.

6.4 أجهزة الإدخال الحقيقية (Physical Input Devices)

نستطيع استخدام مدخلات من أجهزة حقيقية متنوعة لتوليد المدخلات المنطقية لبرامجنا. مع أن البرمجة مع أجهزة منطقية بدلاً من أجهزة حقيقية تتيح للمبرمج الترف وتبعده عن القلق حول موجودية أو خواص الأجهزة الحقيقية المعينة، حيث تكون الاختلافات ما بين الأجهزة الحقيقية المتنوعة إلى حد تؤدي إلى أن يكون أحدها مناسباً بصورة أكثر لتطبيق معين من غيره. عند هذه المرحلة ، سنتحول قليلاً لإلغاء نظرة على أجهزة إدخال أكثر شيوعاً.

1.6.4 لوحة المفاتيح (The Keyboard)

الشكل 9.4 لوحة المفاتيح

توفر لوحة المفاتيح كالمبينة في الشكل 9.4 ، شفرات الرموز القياسية للمنظومة استجابة لضربات المستفيد على المفاتيح. المفاتيح الخاصة، كمفتاح السيطرة تتيح للمستفيد أن يستخدم فئات من الرموز

تصل إلى 256 رمز. أيضاً يستطيع المعالج أن يقوم بترجمة ضربات مفاتيح (Keystrokes) بصورة منفردة أو سلسلة من ضربات مفاتيح بطرق خاصة. على سبيل المثال، الآن معظم لوحات المفاتيح تحتوي على مفاتيح تحمل أسهم (Arrow Keys). تستطيع منظومة الرسومات استخدام هذه المفاتيح ذات الأسهم لتحريك المنزلقة أو للحصول على معلومات موقعية أخرى. في كثير من الأحيان، تشمل لوحات المفاتيح الحديثة المستخدمة في الرسومات دواليب إبهامية (Thumb Wheels) وتجهيزات أخرى لزيادة مرونتها في الاستخدام.

عندما تحدث عملية الضرب على المفاتيح، يتم خزن هوية أو شفرات المفاتيح المضروبة في خازنة مرحلية للوحة المفاتيح (Keyboard Buffer) موجودة في المعالج. يمكن استخدام تمهيد الجهاز (Initialization Of Device) إما بمسح (Clear) أو تحميل الخازنة المرحلية هذه . يتم الحصول على مدخلات نمطي الطلب والحدث من هذه الخازنة المرحلية استجابة لإشارة قدح، عادة يتم توليدها إلا بواسطة مفتاح العودة أو الإدخال (Return Or Enter Key).

تستطيع لوحة المفاتيح تجهيز عدد من القياسات المختلفة، معتمدة على كيفية قيام المنظومة بترجمة أو تفسير محتويات الخازنة المرحلية. فيما يلي بعض هذه التفسيرات:

1- يكون تنفيذ الجهاز المنطقي لصف من الرموز (Logical String Device) بواسطة الترجمة القياسية لشفرات الرموز (Character Codes).

2- باستطاعتنا تنفيذ جهاز الاختيار (Choice Device) بواسطة ترجمة شفرات الرموز إلى مفاتيح رقيمة كعشرة اختيارات أو بتعريف تركيبات من مفتاح سيطرة ورقم (Control Key – Number) للحصول على اختيارات إضافية.

3- أيضاً يمكننا تنفيذ جهاز مقدر أو مخمن (Valuator Device) بواسطة طبع أو إدخال الرقم ببساطة.

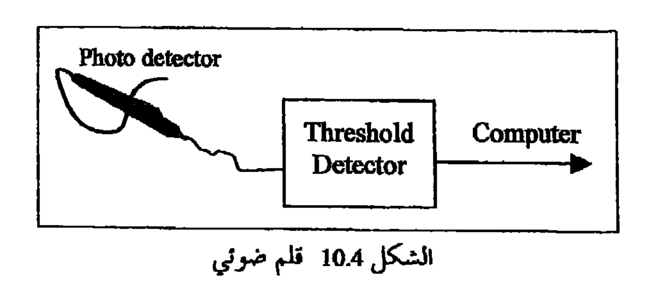
4- يمكن تنفيذ جهازي تحديد الموقع والالتقاط (Locator And Pick) بواسطة استخدام مفاتيح ذات أسهم (Arrow Keys). عندما يعد الجهاز المنطقي للبدء، تقوم المنظومة بتثبيت موقع البداية. وباستخدام المفاتيح ذات الأسهم

يتغير الموقع الذي يمكن إعادة الصدى (Echo Back) إلى المستفيد وذلك بواسطة عرض علامة أو منزلقة.

يحتاج تنفيذ الالتقاط إلى حسابات تؤديها البرمجيات المنفذة أو المكونات المادية. بإمكان لوحة المفاتيح تزويد الموقع، ولكن بعدئذ ينبغي على المعالج إيجاد القطع المطابقة لهذا الموقع. هنا تعتبر مسألة التنفيذ صعبة نوعا ما، معتمدة على مدى القرب المطلوب من المستفيد أن يضع المنزلقة على عنصر موجود في القطعة. التمارين 3.4 و 4.4 تبين طرق لحل هذه المشكلة.

2.6.4 القلم الضوئي (Light Pen)

كان القلم الضوئي من أوائل الأجهزة الخاصة المتوفر للرسومات بالحاسوب المتفاعلة (Interactive Computer Graphics). بالرغم من شعبية الابتكارات الأكثر حداثة، فإن القلم الضوئي لا يزال مستخدماً في عديد من التطبيقات. الشكل 10.4.



الشكل 10.4 قلم ضوئي

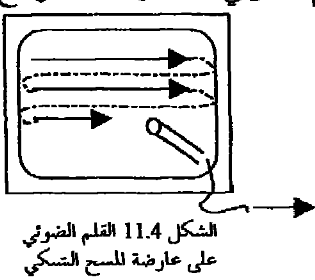
يبين أن القلم الضوئي لا يكتب (أو يبعث) الضوء، لكنه على الأصح يقرئه (أو يتحسسه). لقد تم وضع حد (عتبه) لشدة الضوء (Threshold) الذي يستلمه القلم الضوئي بحيث عندما تتجاوز شدة الضوء هذا الحد تتولد إشارة مقاطعة (Interrupt). يعلم المعالج بالضبط متى يحصل هذا الحدث، لذا يقوم باستخدام هذا التوقيت لحساب موقع القلم الضوئي على وجه أنبوبة الأشعة الكاثودية CRT.

إحدى المشاكل الرئيسية مع القلم الضوئي هي أنه يجب أن يكون هناك ضوء يكفي ليتحسسه. إذن ، لو استخدمنا قلم ضوئي كجهاز التقاط، عادة يتطلب الأمر منا أن نضع القلم على كيان أولي مضاء. فإذا كان هذا الكيان الأولي خط رفيع، قد يواجه المستفيد

صعوبة في وضع القلم الضوئي بدقة علية. هنالك مشاكل أخرى مع القلم الضوئي تتضمن ظاهرة تضائل النصوع (Vignetting– معناه قلة الضوء عند حافات العارضة عما هو عليه في المركز) ومعدل الضوء المنبعث من CRT يضمحل عندما تنقطع إضاءته بواسطة حزمته الإلكترونية.

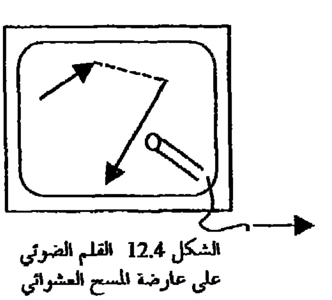
يكون تناول حدث القدح (Triggering Event) مختلفاً إلى حد ما في منظومة المسح العشوائي مما هو عليه في المسح الشبكي. ما يلي هو شرح مختصر لهاتين المنظومتين:

1- تعرض منظومة المسح الشبكي المعلومات خط بعد خط. تبدأ كل دورة إنعاش (Refresh Cycle) عند توقيت دقيق ويواصل العرض خط بعد خط. وبمعرفة متى حصل حدث القدح، تستطيع المنظومة حساب بدقة أين كان موقع القلم الضوئي على مسطع CRT عند ذلك الوقت (الشكل 11.4)، إذن استخدام القلم الضوئي كجهاز تحديد موقع (Locator Device) يكون سهل إلى حد ما. وأما بالنسبة لاستخدام القلم الضوئي كجهاز التقاط (Pick Device) مع منظومة المسح الشبكي، ينبغي على المعالج أن يستخدم موقع على العارضة كي يتم تحديد القطعة.



الشكل 11.4 القلم الضوئي على عارضة المسح الشبكي

2- أما بالنسبة إلى منظومة المسح العشوائي ، يمكن استخدام القلم الضوئي بسهولة جداً كجهاز التقاط. تؤلف العناصر المعروضة في القطع محتوى ملف العرض (Display File) كما في الشكل 12.4.



الشكل 12.4 القلم الضوئي على عارضة المسح العشوائي

في كل دورة إنعاش للعارضة، حيث تبدأ عند وقت محدد، تقوم للمنظومة بالمرور على القطع الموجودة في ملف العرض ويتم عرض جميع الكيانات الأولية. عندما يطلق القدح، نحن نعلم أي قطعة تم عرضها لذا نحصل على قياس الالتقاط (Pick Measure). أما عند استخدام القلم الضوئي كمحدد موقع (Locator) فينتج عنه حسابات إضافية مبيناً على أساس أياً من الكيانات الأولية سببت القدح. على سبيل المثال ، إذا تجاوزت شدة الضوء عن حدها (Threshold) أثناء رسم قطعة خط، يقوم المعالج بعملية استكمال

(Interpolate) بين نقطتي النهاية لقطعة الخط لإيجاد موقع القلم الضوئي عند وقت إطلاق القدح.

يمكننا جعل الأقلام الضوئية تكشف بصورة طبيعية مساحات مظلمة وذلك بمسح سطح العارضة دورياً بواسطة بقعة صغيرة. ونظراً لكون الضوء المنبعث من هذه البقعة أمده قصير، المستفيد سوف لا يرى هذه البقعة ولكن بالإمكان اكتشاف البقعة بواسطة القلم الضوئي.

غالباً ما يتم تركيب الأقلام الضوئية مع أزرار. يستطيع المستفيد تنشيطه إما بدفع القلم الضوئي فوق وجه CRT ، إذا كان الزر على طرف القلم، أو بدفع الزر إذا كان يقع على جانب القلم. يعطي الزر طريقة بديله لتوفير عملية القدح. إذن يصبح التحسس بالضوء هو جزء من عملية القياس.

3.6.4 عصا التحكم (The Joystick)

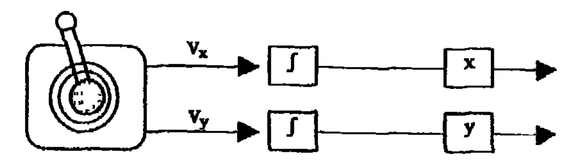
عصا التحكم (الشكل 13.4). عبارة عن جهاز يمتلك درجتين للحرية أو التغيير (Two Degree Of Freedom) أثناء استخدام عصا التحكم القياسية، يقوم المستفيد بتحريك العصا ويؤدي هذا إلى تغير القيم على مقياسين من فرق الجهد (Two Potentiometers). هذا بدورة يزود المعالج بقيمتين مستقلتين حيث يتم خزنهما في زوج من المسجلات (Registers). مثل هذا الشكل التكويني يجعل عصا التحكم جهاز طبيعي لتحديد موقع. عادة يكون تجهيز القدح بواسطة زر موجود على العصا.

الشكل 13.4 عصا التحكم

بالإضافة إلى ذلك، عصا التحكم غالباً تستخدم بأسلوب مختلف بعض الشيء الذي يعطيها فوائد فريدة.

عندما يتم تركيب عصا التحكم مع أجزاء ميكانية كاللوالب قد يعطي المستفيد عند استخدام العصا شعور طبيعي إلى حد ما يشبه عصا السيطرة في طائرة. عادة يتم تركيب عصا التحكم بطريقة تزداد فيه المقاومة كلما قام المستفيد بدفع العصا مبتعداً من موضع استقرار العصا وتعود العصا إلى موضع الاستقرار عندما يحررها المستفيد.

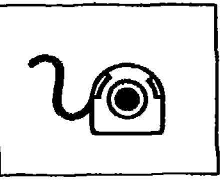
مع الخواص الميكانيكية هذه، عصا التحكم لا تكون ملائمة في استخدامها كجهاز تحديد موقع بصورة مباشرة. لنأخذ بنظر الاعتبار التعديل المبين في الشكل 14.4 .



الشكل 14.4 عصا التحكم كجهاز سرعة

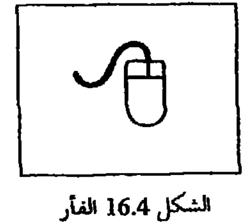
حيث تجري عملية تكامل (Integrated) على مخرجات مقياس فرق الجهد تخزينهما في مسجلات. هذا الشكل التكويني لعصا التحكم يتيح لنا بتفسير مخرجات مقياس فرق الجهد كسرع (Velocities)بدلاً من مواقع وهذه الطريقة يتم تحويل الجهاز إلى جهاز تزايدي أو نسبي (Incremental Or Relative Device). لو افترضنا أن العصا في موضع استقرارها، تكون كلتا السرعتين y,x صفراً، وهذا يؤدي إلى عدم تغيير القيم في المسجلات. بالإضافة ، الجهاز الآن يمتلك حساسية متغيرة. لذلك حركة صغيرة (سرعة واطئة) من موضع الاستقرار قد يؤدي إلى تغييرات بطيئة وصغيرة في محتويات المسجلات ولكن الحركة الكبيرة من موضع الاستقرار يعطي تغيير موقع بسرعة ولكن ليس بدقة.

4.6.4 كرة المسار والفأر (The Trackball And The Mouse)



الشكل 15.4 كرة المسار

الشكل 15.4 يبين كرة المسار التي تكون مشابه إلى عصا التحكم في السيطرة على جهازي فرق الجهد. في هذه الحالة، يقوم المستفيد بتغيير قيم قياس فرق الجهد وذلك بتدوير الكرة. إذن ، كرة المسار عبارة عن جهاز تحديد موقع نسبي، حيث سرعة الكرة تحدد التغيير في قيمة الموقع المخزون في مسجلات المعالج.



الشكل 16.4 الفأر

لو أخذنا كرة المسار وقلبناها رأس على عقب سوف نحصل على جهاز يدعى الفأر (الشكل 16.4) يقوم المستفيد بالقبض على الجهاز ودحرجته على سطح. يوجد على الجهاز زر واحد أو أكثر يتيح للمستفيد بتوليد إشارات القدح. نظراً لكون الفأر جهاز تحديد موقع نسبي، يكون بمقدور المستفيد التقاطه وتحريكه إلى

موقع جديد بدون إعطاء أية إشارة تغيير إلى المعالج. إن جهاز الفأر المبين في الشكل 16.4. هو عبارة عن جهاز فأر الكتروميكانيكي لأن الحركة ميكانيكية (الكرة) وجهاز التحسس هو كهربائي (مقياس فرق الجهد). وهناك فئران شائعة أخرى مثل كهروبصرية وبصرية- ميكانيكية.

لربما يعتبر الفأر أكثر رواجاً بالنسبة لأجهزة الإدخال المتوفرة في الوقت الحالي. لأنها ذات كلفة واطئة ويمكن الاعتماد عليها أو يعود عليها (Reliable) وسهلة الاستخدام. إن تنفيذ عمليتي تحديد الموقع والالتقاط مع الفأر يكون مشابه إلى تنفيذهما مع عصا التحكم لهذين الاستخدامين. مع كل هذه الأجهزة، يكون من الصعوبة الحصول على تحديد موقع مطلق، الذي يكون ضرورياً بالنسبة لبعض المهمات كإدخال بيانات من خريطة أو الرسم باليد.

5.6.4 لوحات البيانات (Data Tablets)


الشكل 17.4 لوحة بيانات

لوحة البيانات المبينة في الشكل 17.4. تقوم بتزويد المعالج بمواقع مطلقة. يقوم المستفيد بوضع ابرة التسجيل (Stylus) على لوحة البيانات. بالنسبة للوحة بيانات نموذجية، تستطيع إبرة التسجيل تحسس إشارات كهرومغناطيسية التي ترسل عبر أسلاك متباعدة بدقة في لوحة البيانات. بالإمكان حل شفة هذه الإشارات لتحديد موقع إبرة التسجيل في دقة متناهية. يوجد هنالك زر أما على الجانب أو على طرف إبرة التسجيل لعملية القدح.

إن كيفية استخدام لوحة البيانات كجهاز تحديد موقع أو جهاز ضربة (Stroke Device) يكون واضحاً. بالمثل كأجهزتنا الأخرى ومع بعض الحسابات يكون بالإمكان استخدامه كجهاز التقاط أيضاً. إن تحديد الموقع المطلق لهذا الجهاز يتيح لنا بوضع بعض مساحات لوحة البيانات جانباً للأغراض الخاصة . على سبيل المثال، يمكننا تكوين جهاز صف الرموز (String Device) بواسطة تخطيط لوحة مفاتيح على جزء من لوحة البيانات. هنالك أجهزة كدفاتر التماس (Touch Pad) وهي مفاهيمياً مشابه إلى لوحة البيانات، لكنها تستخدم أصابع المستفيد بدلاً من إبرة التسجيل. من الممكن وضع شاشات تماس

غير مرئية فوق سطح الـــ CRT، لإتاحة الفرصة للمستفيد أن يعمل بصورة مباشرة مع صورة الشاشة، بحيث تصل مساحة اللمس إلى حجم أنامل الأصابع (Finger Tip).

6.6.4 أجهزة إدخال مرسومة بيانياً (Graphical Devices)

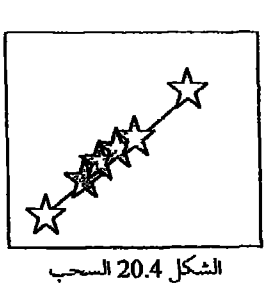
الفرق بين أجهزة الإدخال المنطقية والحقيقية لربما يكون أكثر وضوحاً عندما نستخدم منظومات رسومات لتوليد أجهزة منطقية. يتم التنفيذ الحقيقي لهذه الأجهزة من خلال العارضة وجهاز حقيقي واحد كالفأر. على سبيل المثال، لنأخذ بنظر الاعتبار تنفيذ جهاز اختيار يعمل بثلاثة اختيارات. بالإمكان أن تكون لدينا ثلاثة أزرار فعلية، أو نستطيع استخدام ثلاثة مفاتيح على لوحة المفاتيح أو بالإمكان تكوين "جهاز" كالذي مبين في الشكل 18.4 فيه ثلاثة أزرار ضوئية (Light Buttons). يقوم المستفيد بتعيين اختياره بواسطة استخدام الفأر أو أي جهاز تأشير آخر يكون متوفراً، وذلك بتحريك المنزلقة نحو الزر الضوئي المطلوب والنقر (Click) على الزرق الموجود على الفأر للقدح. أحياناً ما يتم تنفيذ المخمن (Valuator) كما مبين في الشكل 19.4، حيث يكون انزلاق الفأر على امتداد الشاشة للإشارة إلى القيمة المطلوبة . لاحظ كيف نستطيع بسهولة تنفيذ مقياس غير خطي (Nonlinear Scaling) وإلى تنفيذ أي مدى مطلوب مع هذا الجهاز.

الشكل 18.4 أزرار ضوئية

الشكل 19.4 المقدرة البيانية

7.6.4 السحب (Dragging)

في الواقع أن إمكانية استخدام نفس الجهاز الحقيقي كجهازين منطقين مختلفين له نتائج مهمة. بواسطة تقنية تدعى السحب، نقوم باختيار شيء منظور وبعدئذ يتم تحريكه إلى موقع جديد وذلك بجعله يتعقب حركة المنزلقة. أولاً نقوم باستخدام الجهاز الحقيقي كجهاز التقاط لانتقاء شيء منظور. حالاً بعد الانتهاء من عملية الالتقاط، يستخدم نفس الجهاز (بطريقة متكررة) كجهاز تحديد موقع من أجل تعيين موقع جديد للقطعة التي تم



الشكل 20.4 السحب

التقاطها. هكذا تظهر القطعة بأنها موقع من أجل تعيين موقع جديد للقطعة التي تم التقاطها. هكذا تظهر القطعة بأنها تتحرك أو تسحب عبر العارضة وراء حركة المنزلقة كما هو مبين في الشكل 20.4. لربما تحاول إضافة هذه الخاصية لمثالنا المبرمج. إذن بدلاً من إعطاء توجيه للمستفيد أن يدخل معلومات عن الموقع الجديد، يمكنك السماح له أن يسحب نسخة من الشكل المختار إلى الموقع المطلوب. قد يكون عمل هذا الأسلوب أفضل لو استخدمنا نمط العينة للإدخال الذي سوف نبحثه في البند 8.4.

7.4 الالتقاط *(The Pick)*

الآن سنقوم بدراسة الواجهة البينية للمبرمج لأجهزة الإدخال المنطقية. سنأخذ بنظر الاعتبار ثلاثة أجهزة ضرورية لبرنامجنا تخطيط الأشكال وهي:

1- جهاز الالتقاط (The Pick Device)

2- جهاز تحديد الموقع (The Locator Device)

3- جهاز صف الرموز (The String Device)

بالنسبة للمبرمج التطبيقي، يكون استخدام الجهاز في أنماط العينة والطلب متماثلين فعلياً . تكون الإجراءات ومعلميها هي نفسها باستثناء أسمائها- على سبيل المثال، التقاط طلب (Request Pick) والتقاط عينة (Sample Pick). سنستعرض إدخال نمط الحدث (Event-Mode Input) بعد الانتهاء من برنامج تخطيط الأشكال (Layout Program).

يكون جهاز الالتقاط المنطقي عبارة عن جهاز تأشير الذي يعيد رمز تعريف القطعة المشار إليها من قبل المستفيد إلى برنامجه عند حدوث القدح (في نمط الطلب). أيضا يتضمن القياس الحالة المعادة (Return Status) ورمز تعريف الالقاط، الذي يساعدنا في كتابة برامج متفاعلة بسيطة بدون زيادة في تعقيد المكونات المادية.

في برنامج تخطيط الأشكال، أولاً على المستفيد اختيار مفردة من قائمة اختيار الأشكال. نقوم بتصميم البرنامج لكي تعرض الأشكال المتوفرة كشواخص. ويتم وضع هذه الشواخص في قطع، كي تتمكن من انتقاء مفردة بواسطة التأشير إلى أي جزء منها. نحن نرغب أن نعطي المستفيد وقت ليتمكن من وضع المنزلقة على الشاخص المطلوب، الذي يوجه استخدام نمط الطلب للإدخال (Request – Mode Input).

أصبحت لدينا مرونة واسعة، نظراً لكون محتوى القطعة تم تحديده بواسطة البرنامج التطبيقي. تكون إحدى الاختيارات هو استخدام قطعة منفصلة لكل شاخص. مع ذلك، هنالك اختيار آخر، قد يفضل استخدام رموز تعريف الالتقاط. كلاهما ممكن أن يستخدم من خلال إجراء طلب الالتقاط (request pick procedure):

```
void greq-pick (ws – id, pick – num, pick – st, req-pk)
Gint   ws-id;
Ging pick-num;
Gin-st  *pick-st;
Gpick *req-pick;
```

كالمعتاد ws-id هو رمز تعريف محطة العمل الذي يتم تثبيته عند فتح محطة العمل. أما pick-num هو رقم جهاز الالتقاط المنطقي على محطة العمل. قد تمتلك محطة العمل أكثر من جهاز واحد للصنف المعين. لنضرب مثلاً، قد يتم ربط جهاز الفأر على حاسبة شخصية، حيث يمكن استخدام كلا الجهازين لوحة المفاتيح والفأر كأجهزة التقاط منطقية. يتم تثبيت الترقيم موقعياً وربما يجب تحديده لكل تطبيق نقابله، مع أن الرقم "1" عادة يستخدم في تعريف أول جهاز لكل صنف.

1.7.4 استخدام الحالة المعادة (Using The Returned Status)

الحالة المعادة pick – St هي عبارة عن نوع من البيانات قابلة للعدد (Enumerated Type) الذي يعيد أحد القيم التالية GNONE, GOK أو GNO-IN . يتم إعادة نفس نوع الحالة من جميع إجراءات الإدخال وتكون أداة مهمة تساعد المبرمج. دعنا نقوم بتعريف الحالات التالية:

1- إذا كانت القيمة المعادة هي GOK، يعني ذلك أنه تم معالجة الطلب وقدح الجهاز واختيار قطعة.

2- ومن الناحية الثانية، لربما قد تم استخدام الجهاز بصورة صحيحة ولكن لم يشير إلى قطعة قابلة للاكتشاف. في هذا الوضع ، تكون الحالة المعادة GNONE.

3- وأخيراً، قد يحدث شيء ما بينما المنظومة تنتظر عملية قدح الجهاز من قبل المستفيد. على سبيل المثال، يوجد هناك غالباً مفتاح خاص أو زر احتياطي يتيح للمستفيد بالهروب من طلب الإدخال. في مثل هذا الوضع، تكون الحالة المعادة هي GNO-IN التي تخبر البرنامج بأن لم يحدث إدخال صحيح.

نستطيع أن نرى إمكانات مباشرة في استخدام الحالة المعادة في برنامجنا. إذا لم تكن مساحة الرسم داخل القطعة، لذلك لا يمكن التقاط الكيانات الأولية التي تقع في هذه القطعة. لقد تم اتخاذ قرار في استخدام قطع للرسائل لكي نتمكن من إحضارها على العارضة أو حذفها بسهولة من خلال صفة الرؤية (Visibility Attribute). نظراً لعدم وجود أي سبب لالتقاط مساحة الرسالة، يمكننا جعل جميع هذه القطع غير قابلة للكشف. وافترض أننا جعلنا قطع التحكم (Control Segments) أيضا هي غير قابلة للكشف حالياً. لهذا تكون القطع الوحيدة الممكن التقاطها هي تلك التي تتألف منها قائمة الاختيار للأشكال. إذن ، حلقه التكرار للبدء (Initial Loop) قد تكون في الصيغة التالية:

```
do greq – pick (CRT-DEV, PICK-NUM, pick-st, req-pk);
while (* pick-st != GOK);
```

هكذا، تنفيذ البرنامج سوف يتوقف مرحلياً حتى يقوم المستفيد بانتقاء قطعة مرئية. إذا كان كل شاخص له قطعة مستقلة، تكون الخطوة القادمة هي اختيار أي من القطع التقطت لكي تبدأ الفعالية القادمة المناسبة. سيكون هذا الأسلوب ناجح، ولكن قد يكون مربكاً إذا كانت هناك قوائم اختيارات متعددة فعالة ومعروضة في آن واحد. وفي هذه الحالة البديل هو استخدام رمز تعريف الالتقاط.

2.7.4 رموز تعريف الالتقاط (Pick Identifiers)

لنفترض الحالة المعادة تكون GOK، نستطيع تفحص التركيب المعاد المشار إليه بواسطة req-pick. يتكون التركيب المعاد من جزئين:

```
typedef struct {
    Gint  seg – name;
    Gint pick-id;
}Gpick;
```

يكون رمز تعريف القطعة seg-name هو رمز تعريف القطعة الملتقطة. أمـــا رمـــز تعريف الالتقاط pick-id عبارة عن صفة الكيانات الأولية في القطعة. يتيح لنا هذا الرمـــز تأشير الكيانات الأولية الموجودة ضمن قطعة بعلامات مختلفة. عندما نقوم بانتقاء قطعـــة وذلك بالتأشير إلى كيان أولي موجود في تلك القطعة بالإضافة إلى اسم القطعة نحصل على رمز تعريف الالتقاط لهذا الكيان الأولي. إن رموز الالتقاط توفر مرونة إضافية كبـــيرة. في برنامجنا تخطيط الأشكال، يمكننا وضع جميع شواخص الأشكال في قطعة واحدة ويكـــون التمييز فيما بينهما بواسطة تفحص رمز تعريف الالتقاط بعد نجاح عملية الالتقاط. يتـــم تثبيت رمز تعريف الالتقاط بواسطة:

```
void gset- pick-id (pick-id)
```

ضمن تعريف القطعة. حال تنفيذ هذا الإجراء، جميع الكيانات الأولية اللاحقة ضمن هذه القطعة ستحمل نفس رمز تعريف الالتقاط إلى أن يتم تبديلها بواسطة استدعاء إجـراء آخر هو gset – pk-id.

3.7.4 إعداد قوائم الاختيار (Setting up the Menus)

ما يلي برنامج كامل لإعداد قطعة قائمة الاختيار التي تستخدم فيها رموز تعريـــف الالتقاط:

```
static Gpt line-pts [ ]= {{ 0.5; 7.875}, { 1.5, 7.875}};
static Gpt circle –pts [ ]= {{ 1.0, 3.375}, {1.0,2.5}}:
static Gpt text – loc= {1.0,1.125};
static Gtext- align center = {GCENTRE-HOR, GHALF-VERT};
gopen – seg (MENU);
/* Use Previously Set Up
        Normalization Transformation*/
```

(استخدم تحويل معياري سبق تثبيته).

```
gset – norm-tran (MENU-TRANS);
/* line icon: draw  a filled box as backgrond and then line */
```

(شاخص الخط: ارسم إطار مملوء كخلفية ومن ثم الخط).

```
gset – pick-id (LINE);
```

```
fill -box (0.0,2.0,6.75,9.0);
gpolyline (2,line-pts);
/*use two filled boxes for the rectangle icon*/
```

(استخدم إطارين ممتلئين للشاخص المستطيل)

```
gset – pick-id (RECTANGLE);
fill-box (0.0,2.0,4.5, 6.75);
fill-box (0.5, 1.5, 5.0625, 6.1875);
/* circle icon using implementation dependent generalized drawing primitives*/
```

(شاخص الدائرة باستخدام رسم الكيانات الأولية المعممة المعتمدة على التنفيذ).

```
gset-pick-id (CIRCLE);
fill-box (0.0,2.0, 2.25, 4.5);
ggdp (2,&circle – pts, CIRCLE-NUM, NULL);
/ *string icon*/
gset – pick-id (TEXT);
fill-box (0.0,2.0,0.0,2.25);
gset-text- align (&center);
gset-char-ht (1.0);
gtext (&text-loc, "ABC");
gclose-seg ( );
/*make segment detectable*/
```

اجعل القطعة قابلة للكشف

```
gset – det (MENU, GDET);
```

توجد هنا بعض الأمور يجب ملاحظتها:

1- قبل تكوين القطعة، نقوم باختيار تحويل معياري مناسب.

2- بعد غلق القطعة، يتم إعداد القطعة لتكون قابلة للكشف بواسطة الإجراء gset-det. يكون البديل الافتراضي لهذه الصفة طبعاً غير قابل للكشف، والقطعة غير القابلة للكشف لا يمكن التقاطها.

3- أيضاً ينبغي أن تكون القطعة مرئية لكـــي تكـــون قابلـــة للكشـــف، مـــع أن الاحتمالات بالنسبة إلى ما يمكن عمله مع قطع قابلة للكشف وغير مرئيـــة لهـــا جانب من الأهمية.

تكون بعض نواحي استخدام الالتقاط معتمدة على التطبيق. على سبيل المثال، لقـــد استخدمنا ملء كلي لخلفية كل شاخص. من هنا كانت الفكرة هو جعل عملية الالتقـــاط سهلة، نظراً لاستطاعتنا قدح جهاز الالتقاط عند أي موقع داخل الإطار. هذا الأســـلوب قد يكون غير ممكناً مع بعض المنظومات، كأن تكون المكونات المادية غير قـــادرة علـــى اكتشاف ما موجود داخل المنطقة المظلمة . هذه حالة موجودة مع عديد من المنظومـــات التي يستخدم فيها المسح العشوائي مع أنابيب الأشعة الكاثودية CRT'S. قد يكون البديل هو استخدام مساحة ملء كلية مملوءة بألوان مختلفة. وإلا قد يجبر المستفيد بإجراء عمليـــة الالتقاط وذلك بالتأشير إما على الخطوط المحيطة للإطار أو على الشاخص نفسه . الســؤال هنا يكون هو ما مدى القرب الذي يجب أن يشير به المستفيد نحو القطعة (أو عنصر داخل قطعة) لالتقاط تلك القطعة أيضا هذه إحدى الأمور التي تكون معتمدة على التنفيذ.

يتم إعداد قطعة السيطرة (التحكم) بنفس الأسلوب. تحتوي القطعة علـــى رمـــزي تعريف للالتقاط ، حيث يمكننا التمييز بين الشاخص x الذي يستخدم لمســـح الشاشـــة والسهم الذي يستخدم للخروج من البرنامج. لاحقاً في هذا الفصـــل، ســـوف نشـــرح استخدام الشواخص بشكل توضيحي أكبر.

```
static Gpt x – pts-1 [  ] = {{ 0.0,0.0}, {1.0.1.0}};
static Gpt x – pts-2 [  ] = {{ 1.0,0.0}, {0.0,1.0}};
static Gpt arrow – pts – 1[  ] = {{1.5, 0.75},{1.5, 0.25}};
static Gpt arrow-pts – 2 [  ]= {{ 1.25,0.5}, {1.5,0.25};
static Gpt arrow – pts –3 [  ]= {{1.75, 0.5}, {1.5, 0.25}};
gset-norm-tran (CONTROL-TRANS);
gcreate-seg-(CONTROL);
/*the X*/                                (الشاخص x)
gset – pick – id (CLEAR);
fill-box (0.0, 1.0, 0.0, 1.0);
```

```
gpolyline (2x – pts-1);
gpolyline (2x – pts-2);
/* the arrow*/                                  (الشاخص سهم)
gset – pick-id (EXIT);
fill – box (1.0,2.0,0.0,1.0);
gpolyline (2,arrow-pts-1);
gpolyline (2,arrow-pts-2);
gpolyline(2,arrow-pts-3);

gclose-seg (   );
gset-det (CONTROL, GDET);
```

4.7.4 دورة السيطرة (The Control Loop)

الآن تكون دورة السيطرة بأكملها للبرنامج في روعة من البساطة. نقوم بالتقاط والتأكد من صحة اسم القطعة. إذا كان الاسم يشير إلى قطعة السيطرة، معتمداً على رمز تعريف الاتقاط، في هذه الحالة إما نقوم بالخروج من البرنامج أو مسح الشاشة بواسطة إعادة رسم جميع القطع المرئية. إذا قمنا باختيار قائمة الأشكال، يتم استخدام رمز تعريف الالتقاط لانتقاء الإجراءات التي تتيح للمستفيد بأن يقوم بإدخال المعلومات الضرورية تفاعلياً لرسم الشكل المطلوب في مساحة الرسم. تكون عبارات هذا الجزء من البرنامج كما يلي:

```
/* until exit picked */                         (حتى يتم التقاط الخروج)
do {
/* make initial message visible */              (اجعل رسالة البداية المرئية)
gest – vis (INITIAL – MESSAGE, GVIS);
/* request pick until segment selected*/
do greq- pick (CRT-DEV, PICK-NUM, pick-stat, req-pick);
while (pick-stat ! = GOK);
   /* control segment picked*/                  (التقاط قطعة سيطرة)
if (req- pick. seg-name = = CONTROL)
```

```
        if (req-pick. pick-id = = CLEAR)
            gredraw – all-seg-ws(CRT-DEV);
        else exit = DONE;
            /*else menu picked*/
    else switch (req-pick.pick-id);
    {
            case (LINE);
            {
                line (  );
                break;
            }
            case (RECTANGLE);
            {
                rectangle (  );
                break:
            }
            case (CIRCLE);
            {
                circle (  );
                break;
            }
            case (TEXT);
            {
                (text) (  );
                break;
                    }
        }
    }
    while (exit ! = Done);
```

سوف نحتاج إلى محدد موقع ومدخل صف رموز لوضع ورسم الأشكال المطلوبة عند كتابة إجراءات الخط (Line)، المستطيل (Rectangle) ، الدائرة (Circle) والنص (Text). سيتم بحث هذه الإجراءات في البندين القادمين.

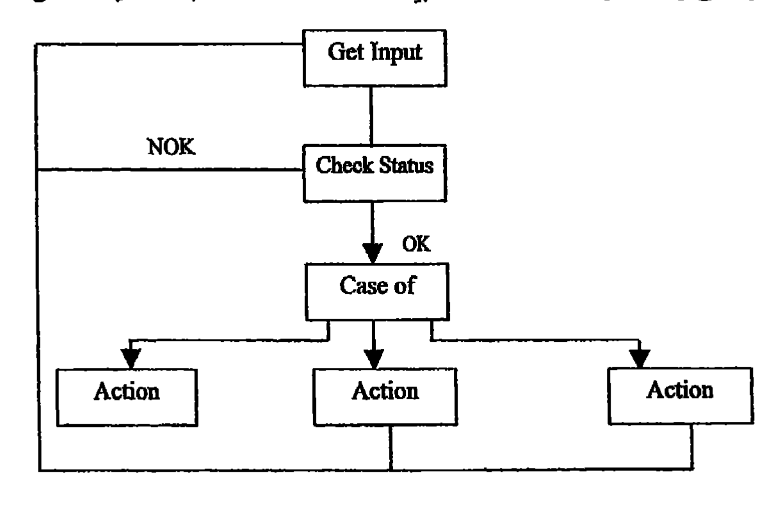
5.7.4 اختيار النمط والإعداد للبدء Mode Selection And Initialization

دعنا نكمل بعض التفاصيل المتعلقة بعملية الالتقاط. يحتوي أمر التقاط العينة gsample-pick بالضبط نفس المعلميات التي يحتاجها الإجراء greq-pick. عادة يمكننا استخدام الجهاز المنطقي في أي من النمطين أو المزج بينهما . يعطينا الإجراء "ثبت نمط الالتقاط – set pick mode" القدرة في التبديل بين النمطين. لقد استخدمنا الالتقاط في حالة بديلة الافتراضي (Default State). يمكن استخدام إجراء "تمهيد الالتقاط the initialize pick" لتثبيت عدد من المعلميات (Paramters) المرافقة مع الالتقاط كنوع الصدى.

إن كثير من هذه المعلميات ستكون مشابه إلى تلك التي تقوم بتمهيد محدد الموقع، لذا سوف نقوم بتأجيل شرحنا حول تمهيد الجهاز (Device Initialization) حتى نشرح محدد الموقع. يكون البديل الافتراضي للصدى والتوجيه (Prompt) على النحو النموذجي هو تعقب الجهاز على الشاشة بواسطة مترلقة واظهار الأجزاء المهمة (High Lighting) أو إيماض (Flashing) القطعة الملتقطة.

6.7.4 الانسيابية العامة للبرنامج (General Program Flow)

تكون انسيابية معظم التطبيقات المتفاعلة مشابه إلى تلك الموجودة في برنامجنا تخطيط الأشكال ويمكن وصفة بواسطة مخطط انسيابي (Flowchart) كالذي مبين في الشكل 21.4.



الشكل 21.4 انسيابية البرنامج

مهما يكون نوع الجهاز، جهاز التقاط أو محدد موقع أو أي من الأنـــواع المنطقيـــة حيث يقوم البرنامج بانتظار مدخلاته، وبعدئذ يستخدم تلك المدخلات لانتقاء فعالية معينة للقيام بها. حالما تنتهي هذه الفعالية يعود البرنامج إلى حالة البداية، وينتظـــر مزيـــداً مـــن المدخلات. من الطبيعي مثل هذا المخطط الإنسيابي ينبغي أن يفسر بالمعاودة أو التكـــرار (Recursively). يمكن أن تبدأ كل فعالية بطلب للإدخال حيث نتائجه بعدئذ تحدد عـــدد من الفعاليات الفرعية (Subactions) المحتملة.

8.4 محدد الموقع *(The Locator)*

افترض أن مستفيدنا اختار شاخص الخط. ســـتظهر الرســـالة "CHOOSE FIRST ENDPOINT OF LINE" على الشاشة، معنى الرسالة هو اختر نقطة النهاية الأولى للخـط. الآن يستطيع المستفيد استخدام محدد موقع للحصول علـــى هـــذه المعلومـــة. في معظـــم المنظومات يستخدم نفس الجهاز (كالفأر) للالتقاط وتحديد الموقع. مرة ثانية، نرغـــب أن نعطي المستفيد وقتاً ليقوم بوضع الجهاز قبل الحصول على قياسه. هنا تكون دالـــة محـــدد موقع طلب (request locator function) مناسبة بدلاً من دالة محدد موقع عينـــه (sample locator function).

1.8.4 طلب تحديد موقع (Request Locator)

يتم الحصول على مدخلات محدد موقع بواسطة هذه الدالة:

```
void greq-loc (ws-id, loc- num, in-st, norm-tran-num, loc-pos)
Gint  ws-id;
Gint   loc-num;
Gin-st *in-st;
Gint  *norm-tran-num;
Gpt  *loc-pos;
```

تكون المعلميات الثلاثة الأولى نفســـها كمـــا هـــي في أوامـــر الالتقـــاط (Pick Commands). نقوم بتعيين ماذا يكون رمز تعريف محطة العمل وأي جهاز علـــى محطـــة العمل نرغب في استعماله. يتم إعادة متغير الحالة (Status Variable). إذا كانت الحالـــة المعادة هي GOK، يمكننا النظر إلى المتغيرين المعادين الآخرين وهما: رقم التحويل المعيـــاري ونقطة في فضاء WC.

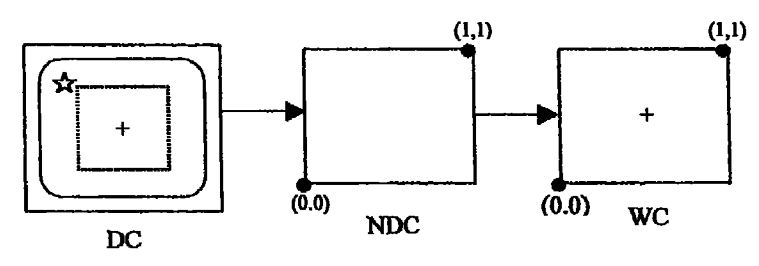
2.8.4 عكس التحويلات الإحداثيات (Inverting Coordinate Transformations)

بالرغم من أن برنامج التطبيق تكون معلوماته المطلوبة في فضاء WC، تكون المدخلات الآتية من الجهاز الحقيقي في فضاء DC عند المستوى الذي تم تجهيزه إلى منظومة الرسومات. يكون من الضروري إجراء تحويلين للإحداثيات قبل استلام البيانات من أي جهاز حقيقي يمكن استخدامه في برنامج تطبيقي. هذان التحويلان هما:

1- إجراء عملية إيجاد معكوس تحويل محطة العمل لنقل بيانات الإدخال من فضاء DC إلى فضاء NDC. وبما أن الطلب للإدخال يقوم بتحديد محطة العمل وجهاز الإدخال المطلوب على محطة العمل، لذا يكون هذا التحويل بصورة واضحة.

2- ينبغي تحويل قيم NDC إلى WC. هنا، نود أن نستخدم معكوس تحويل WC إلى NDC. مع أن في مثالنا، تم تعريف عدد من التحويلات المعيارية . لهذا ، استخدام أي من هذه التحويلات قد يكون غامضاً. الشكل 22.4 يبين لنا أبسط حالة. في مثل هذه الحالة، يكون لدينا تحويل معياري واحد. إذن، لو كان هنالك تحويل واحد فينبغي أن يكون تحويل'0'، نظراً لكونه معرف سابقاً وموجود دائماً. إن هذه النافذة الكونية (World Window) وبوابة الرؤية (Viewport) المعينة تنقل إلى مربع وحده (Unit Square) لكل فضاء NDC. في هذه الحالة، يكون الموقع في فضاء NDC نفسه كموقع في فضاء WC. إن معكوس تحويل محطة العمل يعطي قيمة WC المعادة بواسطة الإجراء greq-loc في المعلمية loc-pos يتم إعادة القيمة صفر في المعلمية norm-tran-num، نظراً لكون هذا هو رقم التحويل المعياري المستخدم في النقل العكسي (Inverse Mapping).

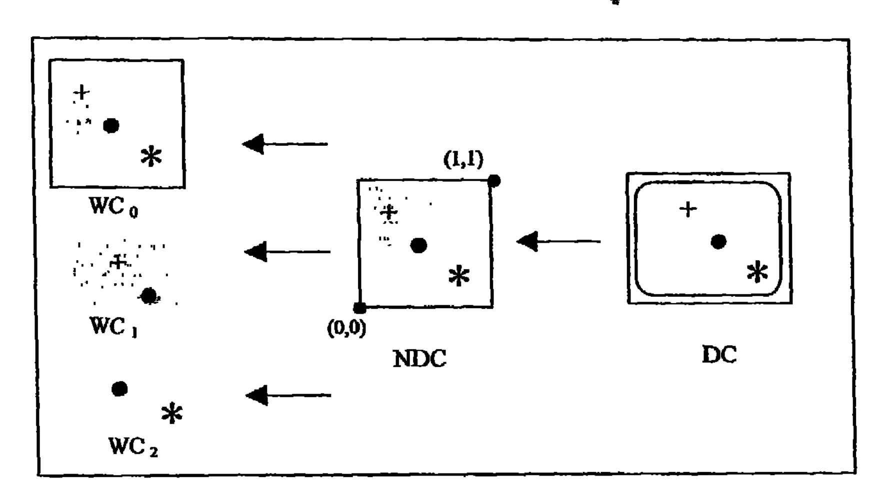
إن هذه العملية تعمل بصورة جيدة مع البقعة الموسومة بعلامة الصليب في الشكل 22.4.



الشكل 22.4 اقتران مدخلات محدد موقع I

ولكن الموقع الموسوم بالنجمة قد يسبب لنا مشكلة. لأن هذه النقطة تقع خارج بوابة رؤية محطة العمل. إنسجاماً مع المتطلبات وذلك بعدم عرض كيانات أولية خارج بوابة الرؤية المخصصة لها، إذن يكون الاختيار المنطقي هو اعتبار موقع النجمة أحد المواقع التي لا يمكن كشفها بواسطة جهاز محدد الموقع. في هذه الحالة يقوم الإجراء greq-loc بإعادة حالة GNONE بعد إطلاق القدح.

الآن لنأخذ بنظر الاعتبار حالة أكثر تعقيداً مبينة في الشكل 23.4. هنا لدينا تحويلان تم تعريفهما من قبل المستفيد، بالإضافة إلى التحويل المعياري "0" المعرف سابقاً. تكون هناك ثلاثة حالات مختلفة وهي:



الشكل 23.4 اقتران مدخلات محدد موقع II

1- يكون الصليب في بوابة رؤية للتحويلين 'o', '1'

2- تكون النجمة في بوابة رؤية للتحويلين '0','2',

3- أما النقطة تكون في بوابة رؤية للتحويلات الثلاثة بأجمعها.

الآن يوجد هنا غموض كبير بسبب الحاجة إلى معرفة مع أي من التحويلات ينبغي الإجراء greq – loc أن يستخدم لتحويل قيم في فضاء NDC إلى قيم في فضاء WC. بالنسبة لمنظومة GKS يجعل هذا الاختيار مبنياً على أساس أسبقية بوابة الرؤية (Viewport

Priorities) . بالإمكان تخصيص أسبقية لكل بوابة رؤية بواسطة الدالة "ثبت أسبقية إدخالات بوابة رؤية" (set viewport input prioritiy):

```
void gset – vp – in –pri (tran – num, ref – tran – num, pri)
Gint tran – num;
Gint ref- tran-num;
Gpri pri;
```

إن أسبقية بوابة الرؤية يتم معرفته بواسطة رقم التحويل المعياري، حيث يثبت عالي (GHIGHER) أو واطئ (GLOWER) نسبة إلى بوابة رؤية مرجعية (Reference Viewport) بواسطة متغير قابل للعد pri. إذن، مع هذا الإجراء، يستطيع المستفيد تثبيت أي ترتيب مطلوب للتحويلات المعيارية.

3.8.4 إدخال البيانات (Entering Data)

في برنامجنا تخطيط الأشكال ، لقد استخدمنا محدد الموقع في مساحة الرسم. وتم تثبيت التحويل المعياري '1' لهذه المساحة (انظر البند 2.4). لنأخذ بنظر الاعتبار العبارات التي تقرأ نقطة النهاية الأولى لقطعة الخط:

```
Gpt points [2], loc-pos;
Gin-stat status;
Gint tran – num;

gset – vis (4, GVIS);
        :
do greq-loc (CRTDEV, LOC-NUM, &status, tran-num, &loc-pos)
while (( status ! = GOK) | | (&tran-num !=1));
```

تحتوي قطعة 4 على الرسالة "اختر نقطة النهاية الأولى للخط" (CHOOSE FIRST ENDPOINT OF LINE. نحن أرغمنا هذه الرسالة بالظهور وذلك بجعل القطعة مرئية، وبعدئذ نستمر بطلب مدخلات محدد الموقع حتى نحصل على تحديد الموقع الصحيح، حيث يتم إعادة رقم التحويل المعياري لمساحة الرسم(1). لاحظ ذلك إن نجاح فعالية تحديد موقع وحدها لا تكفي، لأن توفر رقم التحويل المعياري الذي تستخدمه المنظومة لإعادة

الموقع الآن يستخدم للتأكد من أن تلك هي القيم المطلوبة. وبعد نجاح إكمال هذه الفعالية سيؤدي إلى استخدام نفس قطعة البرنامج من قبل المستفيد للسماح له أن يقوم بإدخال نقطة النهاية الثانية ومن ثم رسم الخط. الآن سننتقل إلى شرح موضوع مهم وهو تمهيد الجهاز (Initialization Of The Device) الذي أهمل لحد الآن.

4.8.4 تمهيد الجهاز (Device Initialization)

إن إعداد محدد الموقع للبدء (Initializing) مهم من أجل تشغيل برنامجنا بشكل كامل. لنأخذ بنظر الاعتبار كيف نستخدم محدد الموقع لتعين قطعة خط. أثناء تنفيذ دالة طلب- موقع تظهر عادة على الشاشة توجيه من نوع معين. دعنا نفترض أن هذا التوجيه هو عبارة عن إظهار المنزلقة. بعدئذ يقوم المستفيد بوضع المنزلقة أينما يرغب وبعدها يقوم بقدح جهاز محدد الموقع. في الطلب الأول الذي تم شرحه، كان هناك موقع ابتدائي للمنزلقة، ويقوم المستفيد بتغير هذا الموقع أثناء تحريكه للجهاز. عندما يتم قدح الجهاز يحدث أحد الأمرين:

1- إما الحالة المعادة ومؤشر التحويل المعياري يشيران إلى أنه قد تم إدخال نقطة نهاية صحيحة وفي هذه الحالة نرغب الانتقال إلى اختيار نقطة النهاية الثانية، أو

2- الحالة المعادة أو أرقام التحويل تشير إلى وجوب إعادة تنفيذ البرنامج للطلب.

في أي من الحالتين، إجراء طلب تحديد موقع اتبعه طلب آخر لتحديد موقع. لو استخدمنا البديل الافتراضي للتمهيد (Default Initialization)، ستعود المنزلقة إلى نفس موقع البداية كل مرة. سيكون لهذا تأثير مزعج وأيضا ضياع في وقت المستفيد، وذلك بسبب استمرار إعادة وضع الجهاز. حيث نفضل ترك المنزلقة بدون تغيير بعد كل طلب وذلك لكي نواصل العمل من هذا الموقع.

إن دالة تمهيد تحديد الموقع (Initialize Locator) التالية:

```
void gint – loc (ws-id, loc – num, init-norm-tran-num, init –loc-pos, pet,
echo-area, loc-data)
Gint ws-id;
Gint loc-num;
```

```
Gint init – norm-tran-num;
Gpt *init-loc-pos;
Gint pet;
Glim *echo-area;
gloc-data *loc-data;
```

تتيح لنا بوضع المنـــزلقة أينما نرغب قبل استخدام الجهاز. بـــالرغم مـــن أن هـــذا الإجراء يحتوي على عدد غير قليل من المتغيرات ولكن بالمقابل نحصل على قدر معقول مـن السيطرة على كيفية استخدام جهازنا. من الطبيعي يكون البديل هـــو الاســتغناء عـــن استخدام التمهيد والاعتماد على البديل الافتراضي.

كالعادة loc-num, ws-id يحددان الجهاز على محطة عملنا. نقوم باختيـــار موقـــع البداية للتوجيه (Prompt) وذلك باستخدام تحويل معياري init-norm-tran-num لنقـــل منـــزلقة عند موقع في فضاء WC وهو init – loc-pos إلى الشاشة . لهذا لو تم اختيـــار هذه القيم الناتجة (المعادة) من الاستخدام السابق لمحدد الموقع، سيبقى موضـــع المنـــزلقة بدون تغيير أثناء انتقالنا إلى طلب قادم لمدخلات محدد الموقع.

المتغير pet يختار نوع التوجيه والصدى (prompt/echo). يكون هنالك عـــدد مـــن الأنواع الممكنة معتمدة على التطبيق. ينبغي على الأقل وجود نوع واحد في كل تطبيـــق. إن التوجيه الأكثر شيوعاً هو وضع المنـــزلقة عند الموضع الحالي لمحدد الموقع. تستطيع هـذه المنـــزلقة استخدام صفات العلامة الحالية أو أن تكون دائماً في نفس الحجم واللون. هنـاك طريقة ثانية هو استخدام ما يسمى خط نطاق مطاطي (Rubber Band Line) وهو عبارة عن قطعة خط مرسوم من موقع البداية إلى الموقع الحالي لمحدد الموقع. تقوم المنظومة بإعـــادة رسم قطعة الخط هذه أثناء قيام المستفيد بتحريك محدد الموقع. أيضـــاً يمكـــن اســـتخدام مستطيل نطاق مطاطي (Rubber Band Rectangle) الذي يكون قسطرة الرئيسي موصل بين موقعي البداية والحالي (Initial and Current Positions). هنـــالك طريقـــة مختلفـــة تتضمن بعرض إطار (Box) على الشاشة محتواه يكون الموقع الحالي للمنـــزلقة.

بالنسبة للأنواع الأكثر تطوراً من أنواع التوجيه والصدى، قد تحتـــاج إلى إدخـــال معلومات إضافية إلى المنظومة. يقوم المستطيل echo-area بتعريف مساحة في فضـاء DC

على الشاشة التي تستخدم للأصداء (echoes) مثلاً يعرض موضع محدد الموقع كرقمين. سنقوم باستخدام مثل هذه المساحة لعرض مدخلات الرموز في البند القادم. أما سجل البيانات loc-data، يوفر طريقة معتمدة على التنفيذ في تزويد أية معلومات قد يحتاجها النوع الخاص من التوجيه أو الصدى، كالألوان لمساحة الصدى أو نوع خاص لخط النطاق المطاطي.

يكون تمهيد جهاز الالتقاط فعلياً مماثل إلى تمهيد محدد الموقع باستثناء ذلك علينا تزويد قطعة البداية ورمز تعريف الالتقاط بدلاً من موضع البداية لمحدد الموقع ومؤشر التحويل المعياري.

9.4 إدخال صف من الرموز *(String Input)*

الآن يمكننا إضافة الأجزاء الأخيرة التي نحتاجها لإكمال برنامجنا تخطيط الأشكال. باستخدام محدد الموقع، يمكننا رسم الأشكال التالية: الخط، المستطيل والدائرة ونظراً لاستطاعتنا بناء جميع هذه الأشكال من خلال تعيين بضعة نقاط في فضاء WC. يحتاج شاخص النص استخدام إدخال صف من الرموز. إذا اختار المستفيد هذا الشاخص ، ستوجه له رسالة تطلب منه بتحديد موقع في مساحة الرسم يوضع فيه النص. يتم هذا الاختيار كالسابق باستخدام الدالتين ginit-loc و greq-loc. عندما تكتمل هذه العملية، يوجه المستفيد بإدخال صف من الرموز. تكون الدالة لطلب صف من الرموز (request string function) التي تلائم لهذا الغرض هي:

```
void greq-string (ws-id, string-num, in – st, req-string)
Gint   ws-id;
Gint   string-num;
Gint – st *in – st;
Gchar *req-string;
```

المعلميات الثلاثة الأولى تؤدي نفس الغرض كما في إجراءات الطلب الأخرى. الطلب الناجح يقوم بإعادة مدخلات صف من الرموز في req-string.

1.9.4 استخدام استعلام (Using And Inquiry)

نحن نرغب في إظهار الرموز التي ندخلها على صف من الرموز (String) أثناء طبعها على جهاز حقيقي مثل لوحة المفاتيح. لهذا الغرض يمكننا استخدام مساحة الصدى (Echo Area). لقد تم بناء شاشة عرض حيث تحتوي على بعض المساحات التي يجب حمايتها، مثلا مخطط الأشكال الذي نقوم بتصميمه بواسطة البرنامج. ينبغي وضع مساحة الصدى في مكان ما بحيث لا تسبب أية مشاكل وتكون بنفس الوقت مرئية بسهولة للمستفيد. بالرغم أن إجراء التمهيد يتيح لنا انتقاء مساحة الصدى، لكن لسوء الحظ بخلاف أي من القيم الأخرى التي نتداولها، مساحة الصدى ينبغي تعيينها في فضاء DC.

عند هذه المرحلة، إما ننهي المناقشة ونقرر الاستعانة بدليل يدوي محلي (Local Manual) لإيجاد القيم الضرورية لعارضتنا الخاصة أو نتعلم إيجاد هذه القيم بطريقة مستقلة عن الجهاز بواسطة إجراء استعلام (Inquiry Procedure). سنقوم باتخاذ الأسلوب الثاني هذا.

تستطيع إجراءات الاستعلام تقديم خدمات لأغراض متعددة، وهذه تتراوح ما بين إيجاد حالة المنظومة (State of the System) إلى إيجاد خواص الأجهزة الحقيقية المتاحة للبرنامج. تقوم دالة استعلام حول أبعاد مساحة العرض (inquiry display space size) بإعادة أبعاد سطح العارضة الحقيقية إلى البرنامج كرقمين حقيقين يمثلان الأبعاد السينية (x) والصادية (y) للعارضة.

نستطيع استخدام هذه القيم لاختبار مساحة الصدى (لإظهار) مدخلات نصوصنا. يمكننا أن نختار جزء صغير من مساحة الرسالة للصدى. إذا كان min هو أصغر القيمتين المعادة بواسطة الاستعلامة، إذن يتم تحديد المنطقة بواسطة:

```
echo – area . x min= 0.1 * min;
echo- area . x max = 0.8* min;
echo-area. ymin=0.05 * min;
echo-area. ymax= 0.1 * min;
```

التي تقع ضمن مساحة الرسالة. لقد وقع الاختيار على استخدام هذه المساحة لأن صدى صف الرموز سيمسح (Erased) ذاتياً بواسطة الرسالة القادمة التي تظهر حال استلام مدخلات النص.

2.9.4 توقف مؤقت أثناء التنفيذ (Pausing During Execution)

يمكننا استخدام نمط الطلب للإدخال (request – mode input) لحل مشكلة ما تنشأ أحياناً كثيرة حتى في البرامج التي لا تحتاج إلى مدخلات. بشكل اعتيادي، البرامج تستمر بالتنفيذ حتى النهاية. باستثناء وجود طلب للإدخال، يستمر البرنامج في عملية التنفيذ. لنأخذ بنظر الاعتبار راسم البيانات ذاتي التدرج الذي تم تصميمه في الفصل الثالث. حالاً بعد رسم الكيانات الأولية المعينة، ينتقل البرنامج إلى إجراء النهاية finish. يقوم هذا الإجراء بإبطال وإغلاق محطة العمل ويمسح سطح العارضة. قد نجد أن هذا الإغلاق قد يحدث حتى قبل أن نعطي فرصة للمستفيد أن يستوعب معلومات الرسم البياني الذي تم عرضه تواً. إن الحل القياسي لهذه المشكلة هو إدخال عبارة بلغة C "getchar ( );" قبل الإجراء finish بالضبط. تكون الفكرة هنا هو لإيقاف تنفيذ البرنامج حتى يقوم المستفيد بالضرب على مفتاح العودة أو الإدخال (Return Or Enter Key). لكن، هذا الأسلوب قد يفشل في تطبيق الرسومات. لو كانت منظومة الرسومات هي المسيطرة على الإدخال والإخراج (كما هي الحالة على الأغلب)، لذلك الإدخال بلغة C الاعتيادي قد لا يعمل. إذن يكون الحل هو استخدام منظومة الرسومات لنفس الغرض. أي إجراء طلب للإدخال (مثال ذلك greq- string) سيقوم بإيقاف البرنامج مؤقتاً حتى إطلاق القدح.

3.9.4 إكمال برنامج تخطيط الأشكال (Completing The Layout Program)

عند هذه المرحلة، يمكننا جمع أجزاء البرنامج كله لتكوين برنامج كامل لتخطيط الأشكال . تم تنظيم البرنامج كسلسلة دوال في شكل عبارات مستعارة كما يلي:

```
main ( )
{
     init ( );
     transforms ( );
     segments ( );
     loop ( );
     finish ( );
}
```

ينبغي أن لا تكون هنالك مفاجاءات. يستخدم البرنامج الرئيسي الدوال التالية:

1- init لإعداد المنظومة للبدء (تمهيد).

2- transforms يقوم بتثبيت جميع التحويلات المعيارية الضرورية.

3- segments يقوم بتعريف جميع القطع للرسائل وقوائم الاختيار وفعاليات التحكم.

4- يتم تناول عمليات التفاعل من خلال loop، حيث تم كتابة الأجزاء المختلفة في البنود الثالثة السابقة. أيضاً يتضمن هذا الإجراء إجراءات قصيرة لرسم الخط ، المستطيل والدائرة والنص.

نقوم بإنهاء البرنامج من خلال الدالة finish بعد اختيار شاخص الخروج (exit) من الدورة (loop).

من الطبيعي، إن هذا التنظيم ليس فقط الوحيد لمثل هذا البرنامج. حيث لم يستفيد من استخدام جميع الإمكانات المتوفرة في منظومة GKS ، أو جميع تلك الإمكانيات المتاحة في منظومات رسومات أخرى. مع ذلك، لهذا البرنامج هيكل يمكنك تعديله أو كتابة برامج جديدة لأغراض أخرى.

10.4 إدخال مدفوع بالحدث *(Event-Driven Input)*

مع كلا النمطين العينة والطلب للإدخال، يتعين على المبرمج أن يحدد من أي محطة عمل ومن أي جهاز إدخال على تلك المحطة ينبغي استقبال المدخلات. قد لا نرغب بتقييد برنامجنا وذلك بفرض اختيار جهاز معين يستخدمه المستفيد. على سبيل المثال، بعض المنظومات لديها جهاز الفأر ولوحة البيانات كلاهما مربوطان بالمنظومة. بالإمكان استخدام أيا منهما كمحدد موقع، ولربما هناك مستفيدون آخرون لديهم اختيارات مفضلة أخرى.

هنالك قضية قد تكون أكثر أساسية وهي، مع مدخلات الطلب والعينة، حيث تكون انسيابية البرنامج أكثر جموداً. على سبيل المثال، بينما نحن في انتظار قدح من طلب تحديد موقع، سيقوم البرنامج إهمال المدخلات الآتية من كل الأجهزة الأخرى. مثل هذه الحالة لا تكون مقبولة في تطبيقات مثل محاكي الطيران أو لعبة الفيديو، حيث في أي

وقت، المستفيد يمكنه أن ينتقي أحد هذه الأجهزة المتنوعة واستخدامها في تجهيز المدخلات.

في حالة إدخال مدفوع بالحدث، بالإمكان استخدام حدث الإدخال للتحكم بانسيابية البرنامج – ويعتبر هذا إضافة فعالة جداً كأداة برمجة. لنأخذ بنظر الاعتبار منظومة تحتوي على عدد من أجهزة إدخال لأنواع منطقية مختلفة ويفترض جميعها مفتوحة كما من قبل. عندما يتم قدح هذه الأجهزة، مقاييسها توضع في تراكيب بيانات (Data Structure) يعرف بطابور الحدث (Event Queue). كذلك الطابور سيحتوي على هوية الجهاز الذي أنتج الحدث ونوع الحدث (محدد موقع أو ضربة "stroke".. الخ) والبيانات الآتية من الحدث. يستطيع البرنامج تفحص الطابور بواسطة دالة انتظار الحدث (await event function):

void gawait-ev (timeout, ws-id, class, in put – num)

إذا كان هناك حدث موجود في الطابور، تقوم الدالة بإعادة المعلومات التالية:

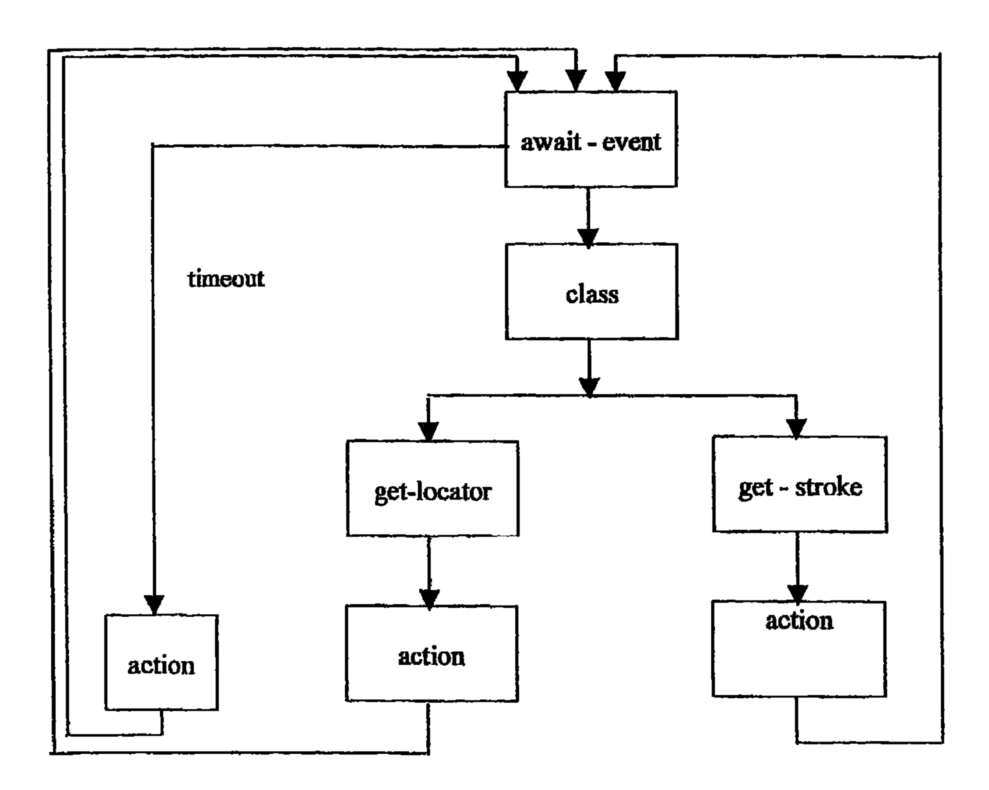
1. أي محطة عمل وأي جهاز إدخال عليها سبب الحدث.
2. إلى أي صنف (GLOC, GPICK, etc.) ينتمي الحدث.

إذا كان الطابور فارغ، ينتظر البرنامج بمقدار timeout ثانية ليحصل حدث ما. إذا لم يحصل أي حدث، المعلمية class تعيد القيمة GNONE ويستمر البرنامج بالتنفيذ.

أما إذا كان الطابور غير فارغ، تقوم الدالة gawait-ev بحذف أول حدث ويستطيع البرنامج استخدام المدخلات المرافقة مع هذا الحدث بواسطة دالة get مثل get pick:

void gget – pick (pick-st, pick)

تكون المعلومات المعادة (الحالة، رقم القطعة ورمز تعريف الالتقاط) تماماً نفسها كتلك المعادة من التقاط عينة وطلب الالتقاط. وباستخدام إدخال الحدث، انسيابية برنامجنا سوف يتبع النموذج المبين في الشكل 24.4.



الشكل 24.4 انسيابية البرامج المدفوعة بالحدث

11.4 الواجهة البينية للمستفيد (The User Interface)

الآن يمكن القول أن لدينا بعض المعرفة حول كيفية استخدام أجهزة الإدخال، لـــذا سنرى كيف يمكننا استخدام التفاعل بصورة مؤثرة. نظراً لكون عملية التفاعل يشترك فيها الإنسان كجزء من منظومة الرسومات، لذا تكون عملية تصميم الواجهة البينية للمســتفيد ذات أهمية حاسمة لتطبيقنا في أداء وظائفه.

يكون تصميم واجهة المستفيد مهم بالنسبة لجميع التطبيقات التفاعلية بالحاســوب، وليس فقط لتلك التي تستخدم رسومات بالحاسوب. ولكن من الناحية الثانيــة، التقــدم الذي حصل في الرسومات في الحاسوب وخاصة في توفر أجهزة الإدخـــال مثــل الفــأر والسرعة التي يمكن فيها توليد مخرجات على عارضات عالية الدقــة (High Resolution) التي أدت إلى ثورة في تصميم واجهات المستفيد. بدون هذه التطورات، ربما لا زلنا نقــوم

بطبع الأوامر بصورة منفردة من خلال لوحة المفاتيح بدلاً من سحب قوائم الاختيار (Pulling Down Menus) والنقر (Clicking) على الشواخص واستخدام أزرار ضوئية (Light Buttons). إذن، هذا البند يتعلق بكيفية كتابة برامج متفاعلة للرسومات وكيفية تطبيق الرسومات بالحاسوب في تصميم واجهة المستفيد.

إن أية واجهة مستفيد جيدة عليها أن تظهر بعض الميزات ، مثلاً:

1- أن تجعل الفعاليات المحتملة واضحة.
2- تقوم بمساعدة المستفيد في قيادته وإرشاده خلال تنفيذ برنامج تطبيقي.
3- يستلزم وجود معلومات توجيهية على العارضة بدون أن تكون محيرة.
4- وجوب توقع أخطاء قد يرتكبها المستفيد وبناء آلية للاستعادة السريعة.

هنالك بعض العناصر الرئيسية التي يتفق عليها معظم المستفيدون المتمرسين ومصممي الواجهات البينية للمستفيد كجزء من الواجهة وهذه العناصر هي:

1. قوائم الاختيار (Menus)
2. شواخص (Icons)
3. التغذية المرتدة للمستفيد (User Feedback)
4. إعانات للمستفيد (User Aids)
5. تخطيط العارضة (Layout)
6. الألوان (Colours)

سنقوم بدراسة كل واحد من هذه العناصر بصورة مختصرة.

1.11.4 قوائم الاختيار (Menus)

على مدى العقود القليلة الماضية تغيرت طريقة التحكم في انسيابية برنامج التطبيق. عندما أصبحت المحطات الطرفية التفاعلية للحاسوب متوفرة والتي أتاحت للمستفيد التفاعل مع البرنامج، أدى هذا التقدم الهائل للحاجة على تحديد انسيابية البرنامج بواسطة تغذية البيانات إلى البرنامج على بطاقات مثقبة أو أشرطة ممغنطة. سابقاً، كان المستفيد يقوم بإدخال البيانات إما عن طريق الأمر (Command Line)، مثل:

run 23.0 37.5 6.5

أو استجابة إلى رسائل حث بسيطة كالآتي:

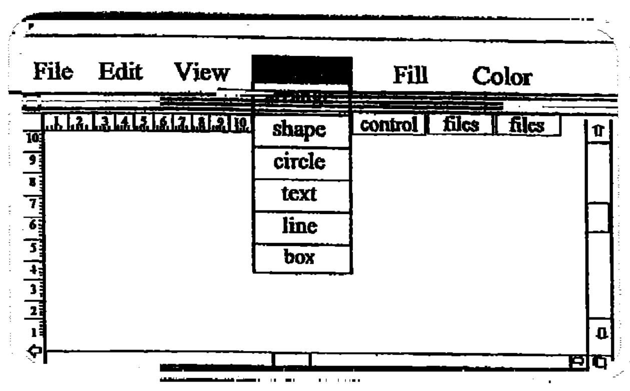
enter number of sides:
enter length in centimeters:
enter 1 to continue, o to exit;

كلتا الطريقتين لها مساوئها. قد تحتاج خطوط من الأوامر إلى معرفة عميقة من قبل المستفيد، في حين النظر إلى سلسلة من الأوامر البسيطة قد تكون مملة بالنسبة للمستفيدين المتمرسين.

لقد نشأت بدائل وذلك بوضع قائمة اختيارات على العارضة . حيث أصبح باستطاعة المستفيد إدخال البيانات والسيطرة على البرنامج من خلال قائمة الاختيار. إن المحطات الطرفية المزودة بمنزلقة تحكم (Cursor Control) أتاحت أن يصبح التفاعل متطوراً إلى حد ما، وحتى بدون قدرات الرسومات. برهنت قوائم الاختيار كبديل يرضي متطلبات مستفيد خبير أو مبتدئ بصورة متساوية. مع عارضة الرسومات، يمكننا تصميم قوائم اختيار تحتوي على نصوص وشواخص بالإضافة إلى ذلك، إن إمكانية الرسومات تعطينا سيطرة دقيقة على وضع القوائم ووقت إظهارها.

في برنامجنا تخطيط الأشكال ، يمكننا تشخيص بضعة طرق في استعمال قوائم الاختيار. في تصميمنا الأولي للبرنامج، لقد أجبرنا المستفيد أن يعمل اختيارات صحيحة من قائمة اختيار وذلك بجعل الفعاليات الأخرى مستحيلة. عندما أردنا من المستفيد أن يختار شكلاً من قائمة الاختيار، جعلنا القطع الأخرى غير قابلة للاكتشاف. يكون البديل هو تغيير العارضة لتشمل فقط قائمة الاختيار للأشكال. نظراً لوجود قائمة اختيار واحدة معروضة، ربما يتمكن المستفيد أن يستغني عن رسائل التوجيه. حالما يتم اختيار شكل، قد تختفي قائمة الاختيار وتظهر مرة ثانية في الوقت المناسب.

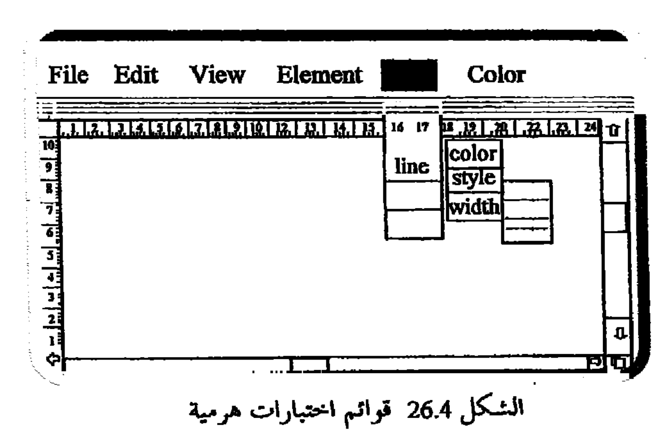
إن إحدى انتقادات هذه الطريقة هي قد يرغب المستفيد أن يقوم بأكثر من فعالية واحدة في كل مرة. وبصورة خاصة، قد نرغب أن نمتلك القدرة على إجراء فعالية سيطرة، كإنهاء البرنامج في أي وقت. فضلا من أن تكون لدينا قائمة اختيار للتحكم (Control Menu) على الشاشة، قد نستعمل شريط سيطرة (Control Bar) كما مبين في الشكل 25.4.



الشكل 25.4 استخدام شريط السيطرة

وكل علامة على شريط السيطرة تشير إلى قائمة اختيار الذي يمكن سحبه للأسفل عندما يقوم المستفيد بالتأشير إلى العلامة. تبقى قائمة الاختيار على الشاشة طالما يكون الزر الموجود على جهاز الإدخال باقياً للأسفل. يقوم المستفيد باختيار مفردة (Item) من قائمة الاختيار وذلك بالانزلاق للأسفل (Sliding Down) عليها ومن ثم تحرير الزر عندما نصل إلى المفردة المطلوبة.

تقدم قوائم الاختيار إمكانات أخرى. حيث يمكن استخدامها هرمياً (Hierachically) على سبيل المثال، افترض أننا نود إضافة صفات كأنواع مختلفة للخط وألوان إلى برنامجنا تخطيط الأشكال. إحدى البدائل هي أن تكون لدينا قائمة اختيار الصفات مبينة على شريط السيطرة . حيث كل مفردة في هذه القائمة يمكن أن تجلب قائمة اختيار أخرى كما هو موضح في الشكل 26.4.



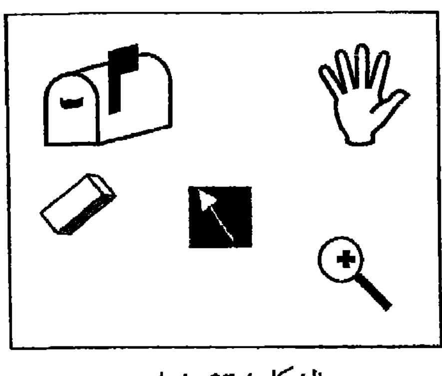
الشكل 26.4 قوائم اختبارات هرمية

إن الاستخدام الفعال لقوائم الاختيار تشمل التفكير بعناية بالغة في محتوى القائمة (أي بمعنى ما عدد المستويات الهرمية التي نرغب بها؟) وأين توضع على العارضة وكذلك ما يتعلق باستخدام الصفات لمساعدة المستفيد. على سبيل المثال، نستطيع بيان المفردات في شريط السيطرة عند إضاءة خافته إذ لم تكن مستخدمة في تلك اللحظة. قد نرغب إظهار (Highlight) مفردة في قائمة اختيار يشير إليها المستفيد بشدة إضاءة أكثر. أيضا قد نرغب توفير اختيارات واسعة لقوائم الاختيار وإعطاء المستفيد إمكانية ترتيب الشاشة مع مجموعة قوائم اختيارات ينتقيها بنفسه. يكون هذا الاختيار بصورة خاصة مهم وذلك قد يستخدم المبتدئ والخبير نفس البرنامج التطبيقي.

2.11.4 الشواخص (Icons)

الشواخص عبارة عن تمثيل صوري للفعاليات أو لأشياء منظورة (Objects). حيث توفر طريقة مرئية للاتصال مع تطبيقنا. ولها فائدة إضافية لكون اللغة الصورية تكون أكثر عالمية من لغتنا المكتوبة الاعتيادية. لربما البرنامج المكتوب والذي يستخدم فقط شواخص للاتصال مع المستفيد يمكن نقله بدون تغيير إلى قطر آخر، وهذا قد لا يكون ممكنا لو كانت قوائم الاختبار مكتوبة بلغة انكليزية أو يابانية.

تكون اختياراتنا للشواخص محدودة لكونها تمثل كصور صغيرة على الشاشة . على سبيل المثال، في كثير من منظومات المسح الشبكي، يكون حجم الشواخص محددة بحيث لا تكون أكبر من 32 × 32 من عناصر الصورة (Pixel). الأشكال البسيطة، مع ذلك (مثل 'x' تشير إلى مسح الشاشة أو سهم يشير إلى الخروج) لا تمثل تلك الفعاليات بصرياً بصورة مباشرة. بعض هذه الشواخص القياسية مبينة في الشكل 27.4.



الشكل 27.4 شواخص

علينا الحذر من افتراض أن الشواخص كالمسح تكون مفهومه عالمياً. قد نتجادل حول استخدام كلمة EXIT كونها توصل معلومات أكثر إلى المستفيد المتحدث باللغة

الإنكليزية من أي شاخص متعلق بفعالية ترك البرنامج. أحياناً، تعرض واجهة المستفيد خيارات في أكثر من طريقة واحدة حيث كثيراً ما تكون مشابهة إلى أسلوب علامات طرق المرور. في الولايات المتحدة ، علامات الطرق تحتوي على ثلاثة أساليب للتعريف: علامات التوقف (Stop) تكون باللون الأحمر (Red Color) وسداسي (الشكل) محتوية على كلمة "STOP" (اللغة) في الداخل، وأما علامة التنبيه فهي عبارة عن مثلثات صفراء اللون تحتوي داخلها كلمة "YIELD" .

أيضاً تكون الشواخص مفيدة للمنزلقات (Cursors). فضلا من وجود منزلقة واحدة كالصليب أو مستطيل صغير، يمكن استعمال للمنزلقات متنوعة التي تساعد في توجيه المستفيد أو تبين له حالة المنظومة (Status of the System). يكون استخدام السهم مساعداً عندما يتوقع من المستفيد دخول موقع، بينما وجهة ساعة (Clock Face) أو ساعة رملية (Hour Glass) غالباً ما تستخدم للإشارة بأن المنظومة مشغولة – وأحياناً تستخدم يد مفتوحة لكي نبين بأن المستفيد حاول إجراء فعالية خاطئة.

3.11.4 تغذية مرتدة للمستفيد (User Feedback)

في جميع التطبيقات سواء أكانت تتضمن على رسومات بالحاسوب أم لا، يكون الاهتمام الأولي للمصمم هو تزويد أكثر ما يمكن من تغذية مرتدة للمستفيد. تقوم قوائم الاختيار والشواخص بتزويد نوعين من التغذية المرتدة. تشتمل الأنواع الأخرى من التغذية المرتدة للمستفيد على ما يلي: إعانات المستفيد (User Aids)، التي سنشرحها لاحقاً، تناول الأخطاء ووسائل مساعدة (Help Facility).

1- إن آلية تناول الأخطاء (Error Handling) التي سبق شرحها لمنظومة Gks تعطينا البداية، ولكن في عديد من النواحي يعتبر هذا تقدم هائل على المنظومات القديمة. في بعض منظومات الرسومات القديمة، إذا حاول المستفيد تحريك المنزلقة خارج الشاشة، قد يسبب انهيار المنظومة بأكملها. إن آلية البديل الافتراضي في منظومة GKS تقوم بتناول هذه المشكلة بواسطة كشف محاولة المستفيد لإدخال بيانات غير مسموحة. يكون رد فعل المنظومة برفض تنفيذ إجراء الرسومات وإدخال هذا الخطأ في ملف الأخطاء. بالرغم أننا قمنا بمنع حدوث انهيار محتمل للمنظومة الذي قد لا يساعد المستفيد

المستخدم لبرنامجنا ولربما يجد أن التطبيق قد تم إيقافه لذا قد يحتاج المستفيد أن يعيد المحاولة مرة أخرى من البداية. مع آلية أكثر تطوراً هو قيام برنامج التطبيق بالتعامل مع الأخطاء. في المثال الذي حاول إدخال بيانات غير صحيحة، كان بإمكان برنامج التطبيق استخدام البيانات أو رسالة الخطأ المولدة بواسطة منظومة الرسومات لتصحيح الخطأ. بالنسبة للبيانات غير الصحيحة باستطاعة برنامج التطبيق المعاودة والطلب من المستفيد أن يقوم بإدخال البيانات من جديد.

2- لو ألقينا نظرة على برمجيات التطبيقات أو حزم برامج CAD أو منظومات التشغيل يبين لنا أهمية توفير وسائل المساعدة (Help Facilities) إلى المستفيد.

غالباً ما يحتاج المستفيد فقط إدخال الأمر "Help" أو "?" لتبدأ وسيلة المساعدة. في تطبيق الرسومات يتم اعتيادياً توفير المساعدة من خلال قائمة اختيار المساعدة والتي قد يشار إليها على شريط السيطرة دائماً. المساعدة قد تتراوح من موجز للأوامر أو الوصول إلى دليل يدوي على الخط (On-line Manuals).

3- هناك مجال آخر للاستفادة من التغذية المرتدة وهو أ ن نبين للمستفيد ما قام به من عمل والسماح له بتغيير فعالياته الجديدة تقريباً. على سبيل المثال ، أثناء قيام المستفيد بالتقاط شيء منظور على الشاشة، عادة نستخدم صيغة معينة لإظهار الأجزاء ذات الأهمية (Highlighting). كترميش المفردة المنتقاة، لإظهار ما تم اختياره من قبل المستفيد. فإذا كان هنالك احتمال أن يرتكب المستفيد خطأ، لربما نفضل إعطاء المستفيد القدرة على تغيير اختياره. احدى الطرق البسيطة لتحقيق هذا هو إظهار الاختيار المشار إليه بشدة إضاءة أكبر والطلب من المستفيد أن يقوم بالإشارة إليه مرة ثانية. إذا كانت أخطاء المستفيد غير محتملة الحدوث، فنحن قد نفضل إضافة مفردة إلى قائمة الاختيار تتيح للمستفيد تتبع فعالية واحدة أو أكثر إلى الوراء (Backtrack).

4.11.4 إعانات المستفيد (User Aids)

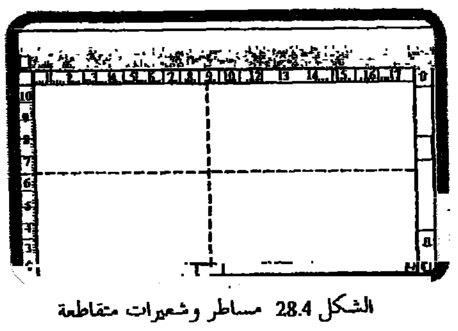
إن إحدى الاستخدامات الرئيسية للرسومات المتفاعلة في التصميم المعزز بالحاسوب (CAD) هي إدخال البيانات من خلال منظومة الرسومات، بدلاً من طبع الأرقام على لوحة المفاتيح أو أية طريقة بطيئة معرضة للخطأ. بدون إعانة الإنسان نحصل على تحديد

موقع ذات دقة واطئة جداً لجهاز إدخال رسومات كالفأر. حيث نكون في حالة توتر وإرباك معاً. لكن، يمكننا استخدام منظومة الرسومات لتعطينا إعانات تساعدنا في التغلب على هذه الصعوبات. تشمل هذه الإعانات على ما يلي:

1- أشرطة منزلقة (Side Bars)
2- شبكات (Grids)
3- تراخ (Relaxation)
4- نطاق مطاطي (Rubber Banding)
5- حساسية متغيرة (Variable Sensitivity)
6- تغذية مرتدة(Feedback)

إن مثل هذه الإعانات فعلاً تكون مهمة لجميع تطبيقات الحاسوب . سوف نقوم بمناقشتها نسبة إلى إدخال معلومات موقعية كما هو الحال في برنامجنا تخطيط الأشكال.

افترض نحن نريد إدخال موقع نقطة نهاية قطعة الخط. في برنامجنا البسيط، يقوم المستفيد بتحريك جهاز كالفأر متتبعاً موقعه من خلال منزلقة على العارضة. عندما يكون المستفيد مقتنعاً بالموقع، يقوم بدفع زراً لتوليد إشارة القدح لإرسال المدخلات إلى البرنامج. لوجود دقة محدودة للعارضة وتوتر المستفيد، يكون الحصول على موقع مضبوط وقابل للتوليد صعب بدون إعانات. تكون إحدى الإعانات الممكنة هو عرض موقع المنزلقة كإحداثيات في بوابة الرؤية على الشاشة.



الشكل 28.4 مساطر وشعيرات متقاطعة

إعانة أخرى قد تكون وضع مساطر مرسومة بيانياً (Graphical Rulers) على جوانب العارضة. غالباً ما تستخدم مثل هذه المساطر بالتضامن مع شعيرات متقاطعة أو متصالبة (Cross-Hair) بدلاً من المنزلقة (كما مبين في الشكل 28.4).

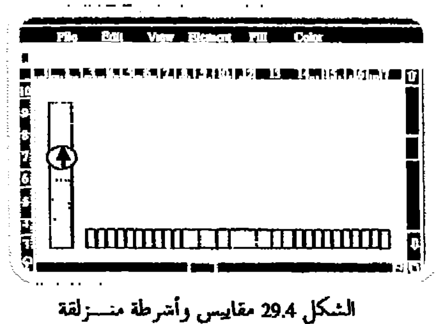
تتيح الشعيرات المتقاطعة حركة معتدلة ومستقلة في الاتجاهين x,y وتكون أكثر نفعاً من منزلقة بسيطة كمؤشر بصري.

تعتبر خطوط النطاق المطاطي بديل آخر للمنزلقات، هنا، يرسم الخط من نقطة بداية ثابتة معينة إلى الموقع المبين بواسطة الحركة لجهاز الإدخال. أثناء حركة الجهاز، يعاد رسم الخط من نفس نقطة البداية إلى الموقع الجديد. هكذا يتغير طول قطعة الخط وأحد نقاط النهاية بأسلوب مرن متتبعاً حركة المستفيد. كيانات أخرى كالدوائر والمستطيلات تستطيع أيضاً استخدام هذه التقنية. على سبيل المثال، يمكن إبقاء أحد زوايا المستطيل ثابتة والزاوية المقابلة قطريا يمكنها تتبع حركة المستفيد.

أحياناً ، وضع مشبك على العارضة قد يساعد المستفيد. في كثير من الحالات، نحن نعلم أن بعض المواقع فقط قد تكون صحيحة. ولربما نعلم أن المواقع تحتاج فقط لقيم ذات أرقام صحيحة لتحديد مواقعها في فضاء WC أو عند فسحات متساوية البعد. لذلك باستطاعتنا مساعدة المستفيد وذلك بتكوين شبكة (قد تكون غير مرئية) ذات دقة مضبوطة. القيم المدخلة بواسطة المستفيد يتم أخذها أو تقريبها لأقرب قيمة للمشبك.

يمكن وضع بعض التقييدات على المستفيد. على سبيل المثال ، يستطيع البرنامج منع المستفيد من تحريك المنزلقة خارج مساحة محدودة على العارضة. مثل هذا التقييد قد يدفع المستفيد إدخال بيانات ضمن مدى صحيح فقط.

إن استخدام منظومة رسومات لتكوين أجهزة مرسومة بيانياً (Graphical Devices) التي تؤدي وظائف أي جهاز من أجهزة الإدخال المنطقية والتي توفر عدة إمكانات جديدة. على سبيل المثال، نستطيع تكوين مقاييس وأشرطة منزلقة (Scales And Slidebars) حيث تتيح تحديد مواقع بصورة دقيقة (الشكل 29.4).



الشكل 29.4 مقاييس وأشرطة منزلقة

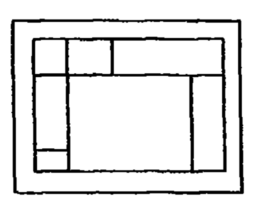
نظراً لأننا نستطيع افتراضياً إيجاد أي اقتران بين موقع على الشاشة والأرقام الحقيقية من خلال البرمجيات، إذن يمكننا إيجاد مقاييس ذات حساسية متغيرة. حيث يمكننا

استخدام نهايتي المقياس في تحديد مواقع بصورة تقريبية واستخدام المركز للتضبيط الدقيق (Fine Tuning).

هناك إعانة أخرى مهمة هي السماح للمستفيد بالعمل مع مقاييس مختلفة في مساحة العمل من خلال التكبير بالتقريب أو التزويم (Zooming). حيث يكون بالإمكان التبديل بين المشاهدات الاعتيادية والتزويم لكي يستطيع المستفيد أن يضع مخططاً للتصميم بصورة تقريبية ومن ثم يضع التصميم بصورة أدق عند الضرورة. مثل هذه التقنية تكون نافعة بصورة خاصة مع التصاميم الكبيرة أو حيثما يكون لدى المستفيد عارضه صغيرة محدودة الدقة.

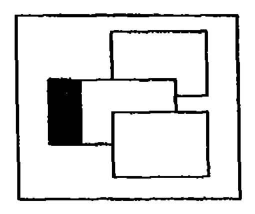
5.11.4 تخطيط العارضة (Layout)

إن نجاح الواجهة البينية للمستفيد تعتمد أيضاً على مظهر رؤيتها. واجهة المستفيد الجيدة ينبغي أن تكون نافعة وغير مربكة للمستفيد. من جانب، نحن نرغب تجهيز المستفيد بجميع المعلومات التي يحتاجها، ومن الجانب الآخر ، إننا نرغب ترك مساحة معينة للمستفيد يعمل فيها. إن استخدام شريط سيطرة كما سبق شرحه، يكون أحد التقنيات في توفير مساحة على سطح العارضة.



الشكل 30.4 تبليط العارضة

إن الأسلوب الذي تم استخدامه لعارضتنا في برنامج تخطيط الأشكال هو استعمال مستطيلات ثابتة الحجوم أو بلاطات (Tiles) التي لا تتداخل. الشكل العام لمثل هذه الواجهة مبينة في الشكل 30.4 . كل بلاط يتطابق مع بوابة رؤية في برنامجنا. يكون البديل هو السماح لبوابات الرؤية إن تتداخل (الشكل 31.4). لاحظ، لسوء الحظ أن كثير من جمهور محطة العمل تستخدم مصطلح النافذة بدلاً من بوابة الرؤية في هذا المضمون. الآن تكون قوائم الاختيار مختبئة الواحدة وراء الأخرى، ولكن إذا سمحنا بوجود شريط علوي مرئي على كل قائمة اختيار، سيكون المستفيد مدركاً أية قائمة اختيار تكون متوفرة . عندما يريد المستفيد استخدام قائمة اختيار معينة، يقوم بالنقر على الجزء المعروض من بوابة الرؤية وهذا سيؤدي بالمنظومة عرض قائمة الاختيار المنتخبة وذلك بجلبها إلى الأمام. مثل هذه الأسلوب لا يتم تطبيقه على قوائم الاختيار فقط. يمكننا استخدام بوابات رؤية متداخلة



بوابة رؤية متداخلة

لإعطاء المستفيد الفرصة للعمل مع رسوم بيانية متعددة في آن واحد. لا يزال هنالك نقاش كثير حول أي من هذه الصيغ تكون أفضل. بالتأكيد الجواب يعتمد على التطبيق، وأيضا على التنفيذ. على سبيل المثال، رسم بوابات رؤية متداخلة قد يتطلب من المنظومة عمل أكثر بالمقارنة مع البلاطات. لا يمكننا تحديد فيما إذا كان ممكننا عمل هذا بدون تباطؤ سرعة تنفيذ برنامج التطبيق إلى درجة حيث لا تكون هناك فائدة إلا الأخذ بنظر الاعتبار الخوارزميات والمكونات المادية لتنفيذ الواجهة البينية .

6.11.4 الألوان (Colors)

قبل أن نترك موضوع الواجهة البينية للمستفيد، سوف نقوم بالتعليق على مدى تأثير اللون على واجهة المستفيد. سنؤجل الشرح المفصل حول الألوان إلى حين بحث موضوع المسح الشبكي في الفصل السابع.

لقد أخذت تقنيات عارضات CRT بالتقدم إلى حد أنها لا تكون أكثر كلفة أو أقل دقة بالمقارنة مع عارضات أحادية اللون (Monochrome Displays) . تكون الألوان مفرحة وإعلامية (Informative) للمشاهد معاً. نستطيع استخدام اللون في منظومتنا للرسومات ببساطة وذلك بإعداد جداول للألوان. إن الاستخدام المؤثر للألوان من ناحية يعتمد على المظهر السايكولوجي لرؤية الإنسان.

إحدى المشاكل الرئيسية مع استخدام الألوان هو الإفراط باستخدامها مع أن، قليلاً من الألوان قد تستخدم لتحسين الواجهة، ولكن استخدام أكثر من خمسة أو ستة ألوان تميل إلى الهاء وتشويش المستفيد. الإنسان يفسر الألوان بأسلوب حراري (Thermal). لكن تفسير الألوان الباردة (مثل الأزرق والأخضر) كموصلة للمعلومات تكون أقل أهمية من تلك الألوان الحارة (مثل الأحمر والأبيض والأصفر).

هنالك عوامل أخرى تتحكم باستخدام الألوان معتمدة على الخواص المادية لآلية الحس في العين. على سبيل المثال، العدسات في العين لا تكون مصححة للألوان كما هو الحال مع عدسات آلة التصوير. لا يمكنها التركيز آنياً على الذبذبات المنبعثة من نهايتي الطيف المرئي معاً. إذن عرض أشياء منظورة متجاورة وباللون الأحمر والأزرق قد تسبب مشاكل للمشاهد. بالإضافة، يكون الموقع الرئيسي لمجسات اللون الأزرق في العين عند محيط الشبكية. لذا يفضل تحاشي استخدام اللون الأزرق كخلفية لأشياء منظورة صغيرة كالنصوص.

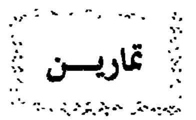
12.4 أعباء التفاعل (The Burden of Interaction)

لقد تعرفنا على آلية التفاعل في منظومة الرسومات وقمنا بدراسة الكيفية التي نستطيع فيها تصميم برامج تفاعلية. نحن لم نقم بمناقشة القضايا المتعلقة بالتنفيذ. قد يكون للتفاعل معنى ضئيل لو لم نستطيع بناء منظومة يمكنها أن تتفاعل مع المستفيد في الزمن الحقيقي (Real-time). بالنسبة لمستفيد التطبيقات الذي يعمل مع برمجيات مستقلة عن الجهاز، يكون من السهل التغاضي عن الأعباء الهائلة التي تضعها الرسومات المتفاعلة على المكونات المادية.

لنأخذ بنظر الاعتبار تفاعل مع منظومة تعتمد على قائمة اختيار تستعمل فيها بوابات رؤية متداخلة. في كل مرة يسحب المستفيد قائمة اختيار للأمام، ينبغي إعادة رسم جزء مهم من الشاشة. مع عارضة المسح الشبكي، يستوجب ملء مساحات كبيرة. إذا تم استخدام وسائل إعانة كشعيرات متقاطعة وخطوط النطاق المطاطي، ينبغي على المنظومة محاولة رسم ومسح الخطوط أسرع من إمكانية المستفيد عند تحريكه جهاز الإدخال. إن حركة الأشياء المنظورة حول الشاشة بواسطة السحب (Dragging) هي حالة أخرى حديث يتم فيها تحديث العارضة بصورة مستمرة تقريباً.

هنالك قضية لها علاقة وهي فيما إذا كانت حزم برمجيات الرسومات القياسية (Standard Graphics Packages) تستطيع إسناد هذا النوع من التقنيات التفاعلية التي تم بحثها. بالتأكيد، عديد من البرامج التفاعلية يمكن تطويرها مع برنامج المستفيد مستخدمة منظومتي GKS و PHIGS. توجد هنالك حالات أخرى حيث لا يكون واضحاً من أن التفاعل ممكناً. على سبيل المثال، برنامج مستفيد يعتمد على كيانات أولية ذات المسح العشوائي (أشكال مرسومة بخطوط)، كما في منظومتي GKS و PHIGS، قد لا يمكنه تكوين خط نطاق مطاطي.

عند دراسة منظومات موجودة حالياً يتوضح لنا أنه بالإمكان بناء منظومات عالية التفاعل. وبدراسة قضايا التنفيذ، سوف نتفهم بصورة أفضل أين تكمن التحديدات وسنكتشف أي من أجزاء التفاعل ينبغي أن تكون جزء من البرنامج التطبيقي وما يجب أن تكون مسؤوليات المنفذ. من الآن، ينبغي أن تكون قادراً للبدء في كتابة برامج تفاعلية والقيام بتجارب مع شكل وأسلوب الواجهات البينية للمستفيد.



1.4 لنأخذ بنظر الاعتبار العمليات الأربعة مع القطع وهي: استحداث (Create)، غلق (Close)، حذف (Delete) وإعادة التسمية (Rename). قارن عملية التنفيذ بين استخدام قوائم موصلة (Linked-List) واستخدام الجداول. ناقش مع كل نوع من التنفيذ مشاكل إدارة الذاكرة (Memory-Management) والتوقيت.

2.4 كيف تقوم بعملية إضافة تعديل قطعة (Segment Modification) أو وسيلة تنقيح (Editing Facility) إلى حلك للتمرين 1.4.

3.4 لنأخذ بنظر الاعتبار تنفيذ جهاز التقاط بواسطة جهاز حقيقي، مثل لوحة مفاتيح أو لوحة بيانات التي يمكن أن توفر موقعاً. تكون إحدى الاحتمالات هو جعل إطار تحديد يترافق مع كل قطعة . احتمال آخر هو استخدام قائمة عرض (Display List). بالنسبة لكل حالة، حاول أن تجد طريقة في تحديد رمز تعريف قطعة.

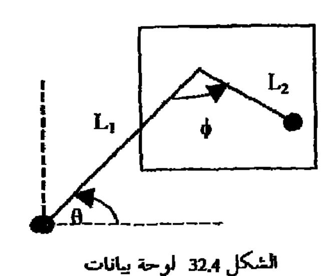
4.4 إن تعريف مدى قربنا لكيان أولي في قطعة من أجل التقاطه متروك إلى مواصفات معظم منظومات الرسومات. في ضوء تمرين 3.4 وما تعرفه عن أجهزة الإدخال والإخراج المطلوب أن نصل إلى تعريف مقبول في استخدام جهاز حقيقي معين: هل يؤثر على ذلك فيما إذا كانت العارضة من نوع المسح العشوائي أو الشبكي؟

5.4 لنأخذ بنظر الاعتبار عارضة CRT قياسية كما تم وصفه في الفصل الأول. افترض حزمة الكترونية تم تركيزها بصورة مثالية داخل عمود سعته w. عند مركز الشاشة CRT، سنرى بقعة سعتها w. كيف تتغير سعة وشدة الضوء أثناء حركة الحزمة عبر سطح CRT؟ كيف يؤثر هذا التغيير على انتقاء حدة الإضاءة (Threshold) لجهاز القلم الضوئي؟

6.4 فسر ذلك، إذا استخدمنا عصا تحكم كجهاز للسرعة، يمكننا أن نحل محل مقاييس فرق الجهد بزوج من مفاتيح ثلاثية المواضع (Three Position Switches).

7.4 يمكننا القيام ببناء لوحة بيانات ميكانيكية كالمبينة في الشكل 32.4. يتم تحسس مواقع الزاويتين بواسطة معالج رسومات. أوجد موقع ابر التسجيل (Stylus) بدلالة هذه الزوايا

(وطول الأذرع). كيف يتم تغيير حساسية هذا الجهاز أثناء قيامنا بتحريك إبرة التسجيل على السطح؟



الشكل 32.4 لوحة بيانات

8.4 اكتب إجراءات لتنفيذ أجهزة إدخال منطقية مرسومة بيانياً، كجهاز اختيار مخمن (Choice And Valuator). كيف يتم توفير التوجيهات والأصداء (Prompts And Echoes)؟

9.4 مثال برنامج تخطيط الأشكال استخدم إدخال نمط الطلب فقط، يمكن أن نحصل على برنامج أكثر جاذبية وذلك باستخدام إدخال من نمط العينة أو نمط الحدث. أعد كتابة البرنامج مستخدماً أحد أو كلا هذين النمطين.

10.4 لو أعطيت لوحة مفاتيح فقط ، كيف يمكنك استخدامها لتوفير جميع صنوف الإدخال المنطقية.

11.4 تغيير في برنامج تخطيط الأشكال يحدث في رسم مخططات الدوائر المنطقية . لقد تم التعرف على البوابات المنطقية الأساسية في تمرين 9.3. يمكنك توسيع ذلك التمرين إلى برنامج تفاعلي لتصميم الدوائر المنطقية. قد ترغب بإضافة قدرة تحليلية حيث يتم تحديد المعادلة البولينية (Boolean Equations) للدائرة المصممة والقيام بتكوين جدول الصواب (Truth Table).

12.4 المتطلب الذي يجعل نسبة مربع الأقصى (Aspect Ratio) لنافذة محطة العمل وبوابة رؤية محطة العمل لكي تكون نفسها هو استخدام أكبر مساحة مربعة على عارضة مستطيلة الشكل. قد يؤدي هذا الاختيار في فقدان مساحة لا بأس بها على أجهزة الإخراج.
باستخدام إجراء استعلام سعة مساحة العارضة (inquiry display space size)، وضح كيف، لنافذة كونية معطاة، يمكننا اختيار بوابة رؤية كونية، نافذة محطة عمل وبوابة رؤية محطة عمل لاستخدام العارضة بأكملها. هل هناك أي تأثيرات جانبية غير محبذة؟ إن كانت هناك فاشرحها.

13.4 إن لعبة الشطرنج وأخذ المربعات يتم تداولها على لوحة متكونة من 8x8 من المربعات السوداء والبيضاء. يمكن لهذه اللعبة أن تحاكي (Simulated) بواسطة رسومات الحاسوب. وذلك يقوم مستفيدين بتحريك قطع بواسطة جهاز الإدخال. اكتب مثل هذا البرنامج. برنامجك عليه أن يتحقق من صحة الحركات أيضاً.

14.4 باستخدام إجراءات إدخال قياسية التي تم شرحها، استخرج إجراءات إدخال وذلك لإدخال بيانات مع محدد موقع. بحيث يتضمن عمليات دفع البيانات لكي تقع على شبكة محددة من قبل المستفيد.

Pseudocolor enhanced display of autoradiograph showing flow across membranes in the brain. (Courtesy of Departments of Computer Science and Neurology, University of New Mexico.)

Visualization of complex cube root function. (Courtesy of Ardent Computer Corp.)

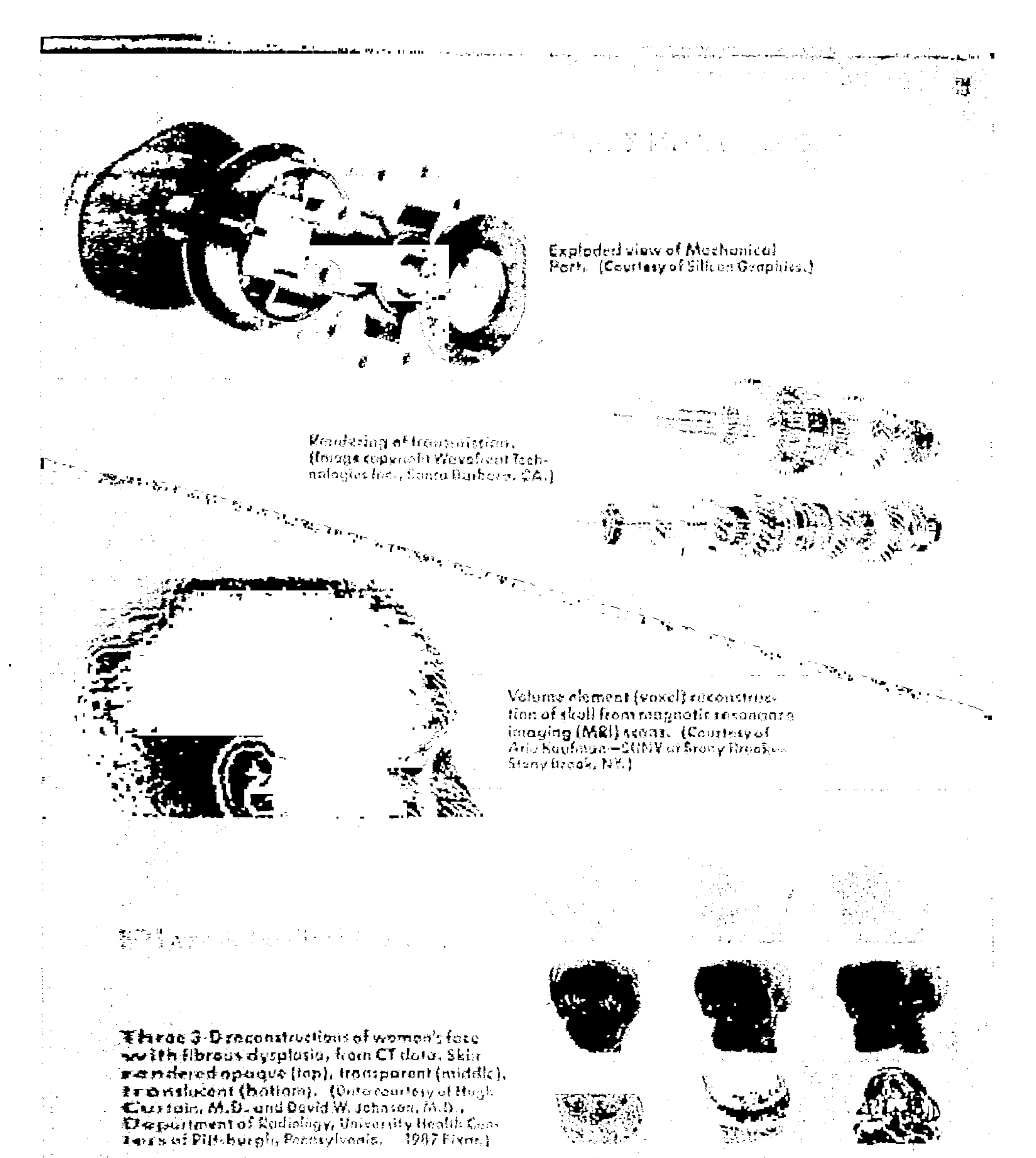
Plate 1 Plotting

Three-dimensional display of data plotted using commercially available software. (Produced by Computer Associates International, Inc.)

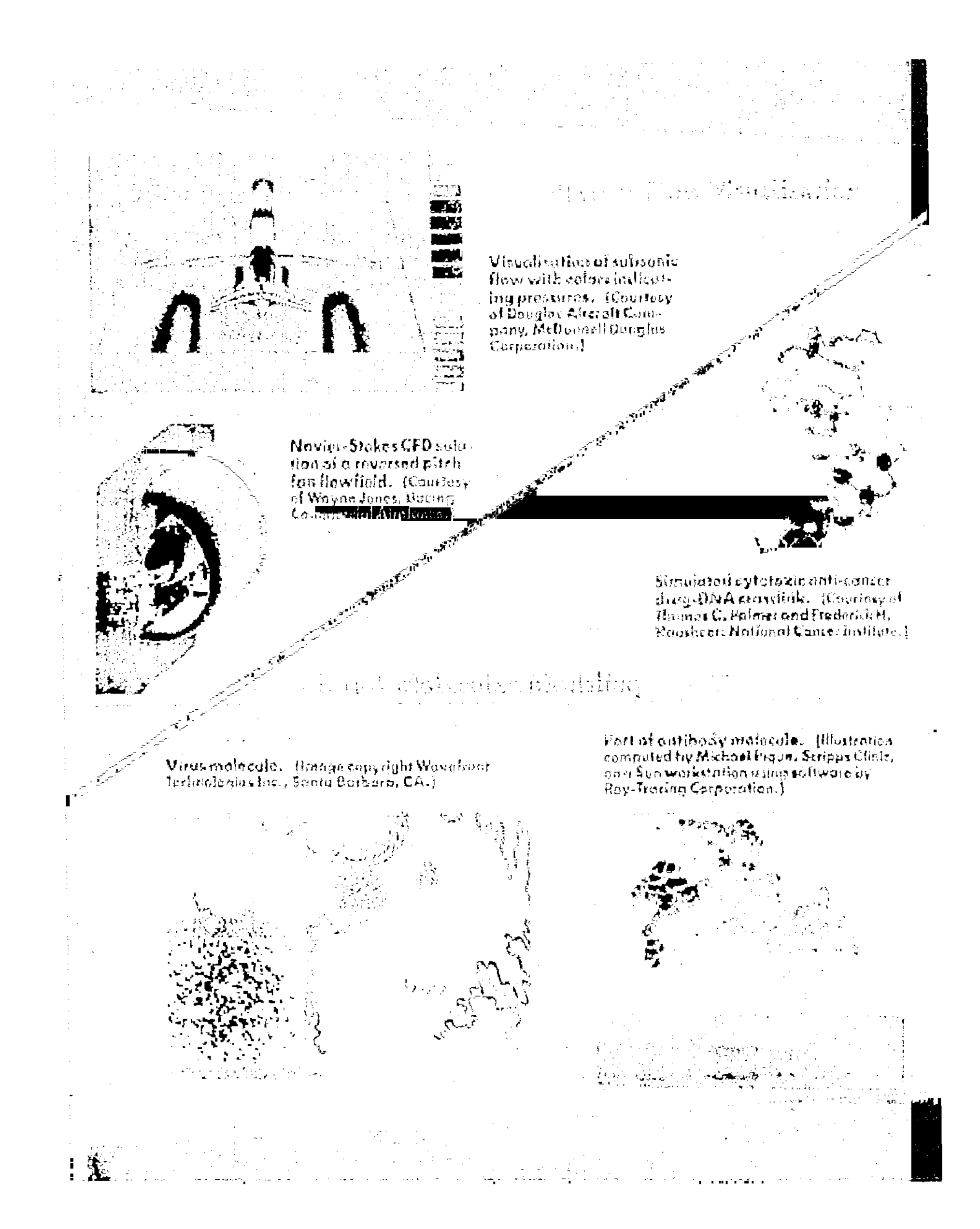
Plate 2 Architecture

Computer renderings of computer-aided architectural designs. (Copyright 1987 and 1989 S. Kasahara-Kajima Corp., Japan.)

الصورة رقم ٢،١



الصورة رقم ٤٠٣



الصورة رقم ٦،٥



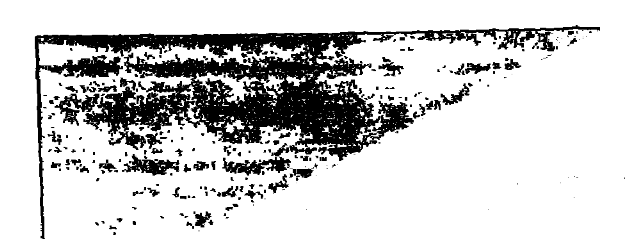
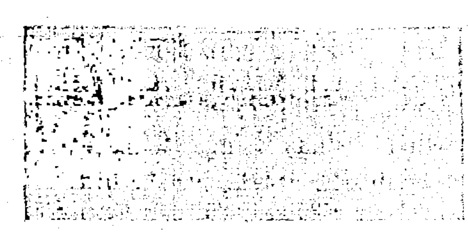
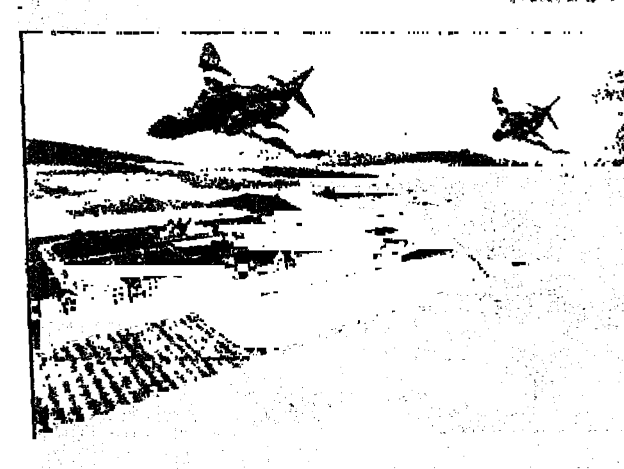


Plate 7 Circuit Design



Computer-aided VLSI design for square root function. *(Courtesy of Imperial College, London.)*

Plate 8 Flight Simulator



Rendered polygonal model of aircraft used in flight simulator. *(Courtesy of Evans and Sutherland Computer Corporation.)*

"صورة رقم ٨،٧



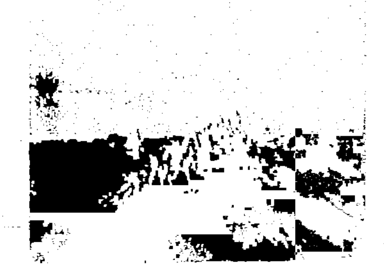


Tricycle. (Courtesy of Gary Mundell and Andrew Pearce, Alias Research.)



Ford concept car. (Image copyright Wavefront Technologies Inc., Santa Barbara, CA.)

Plate 9 Photorealism

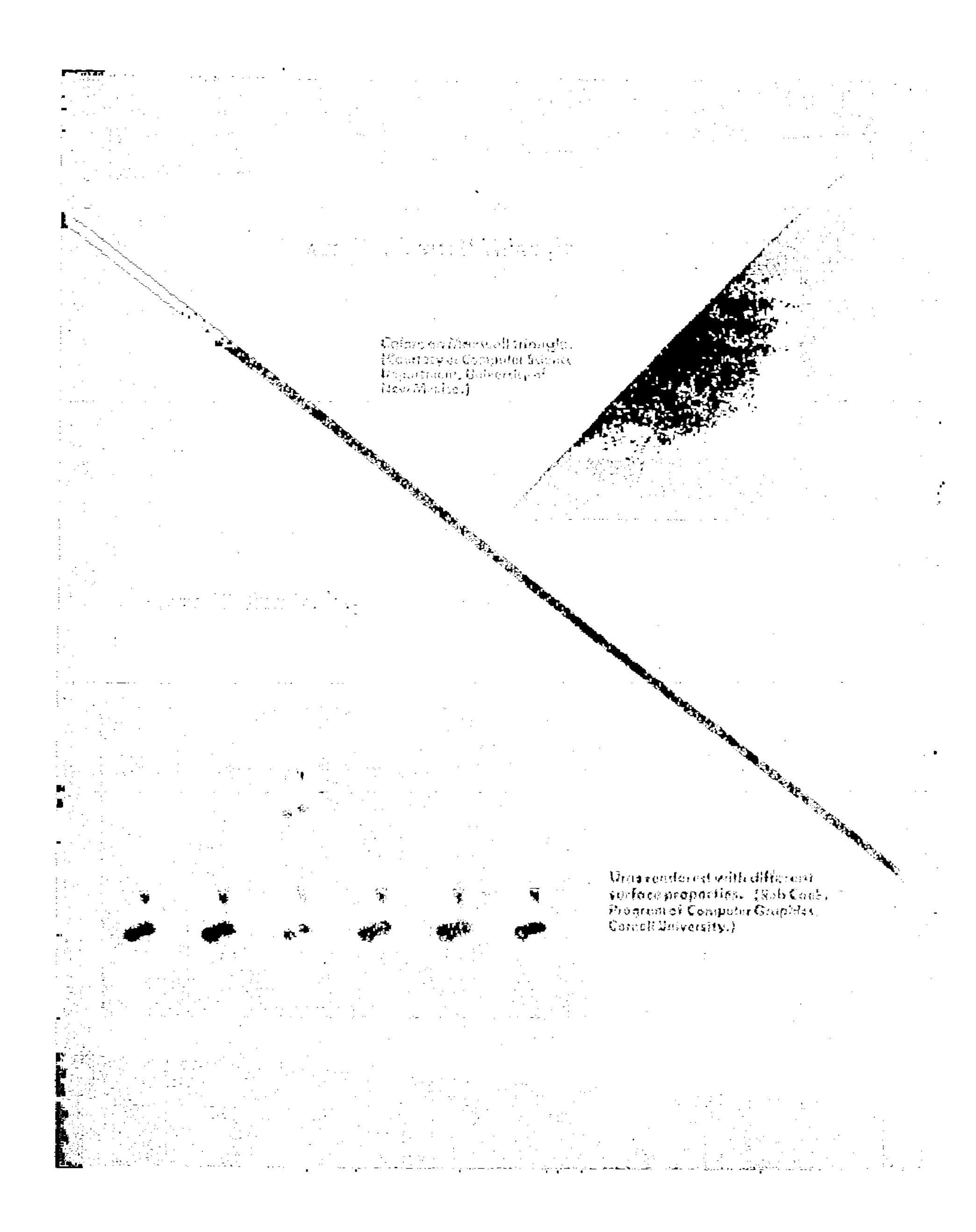


Scene modeled with polygonal terrain and quadric clouds. (Courtesy of G. Gardner, Grumman Data Systems.)

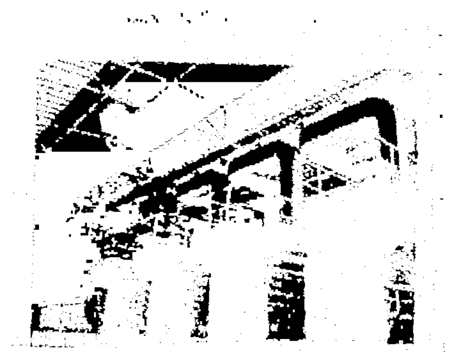
Plate 10 User Interface

User interface during generation of diagram from this book.

الصورة رقم ٩،١٠



الصورة رقم ١١، ١٢



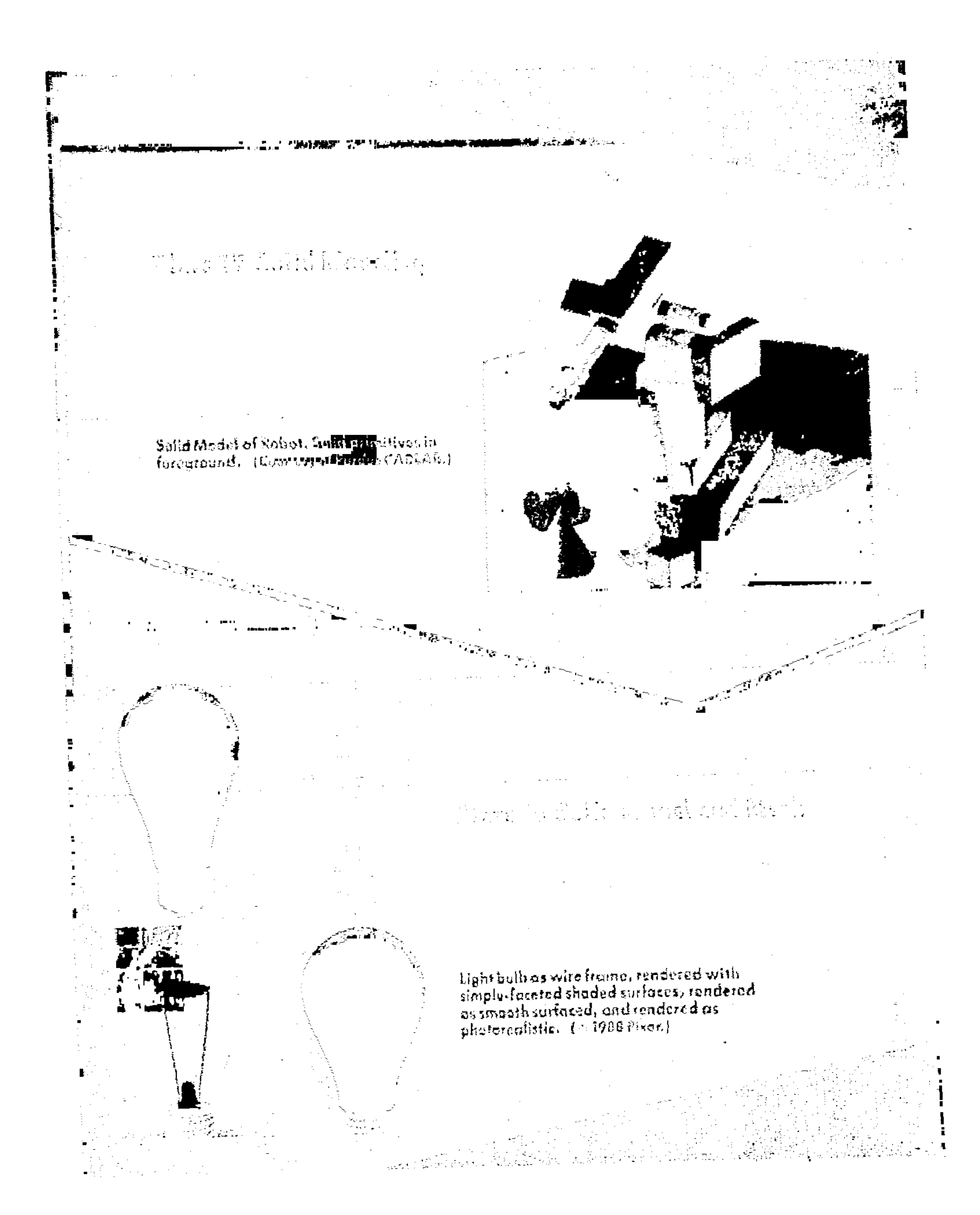
Steel Mill. Illumination using radiosity method. (Stuart I. Feldman and John R. Wallace, Program of Computer Graphics, Cornell University.)

Candle. (Image copyright Wavefront Technologies Inc., Santa Barbara, CA.)

Water Strider. Rendered using the equivalent of 50,000 polygons and 12 different textures. (This image was created by Daniel Langlois, using the SOFTIMAGE 4D Creative Environment software. Copyright 1988 SOFTIMAGE Inc.)

Light Bulbs. Rendered using 793,000 polygons. ("Wire Guy" © 1987 James Dixon/PDI.)

الصورة رقم ١٤،١٣



الصورة رقم ١٥،١٦

الفصل الخامس

التحويلات والنمذجة (Transformation and Modeling)



الفصل الخامس

التحويلات والنمذجة (Transformation and Modeling)

مقدمة:

إلى درجة كبيرة القدرة الكاملة لمنظومة الرسومات الحديثة تستند على قدرة المنظومة في تنفيذ تحويلات على أشياء منظورة بيانياً. لقد قمنا باستخدام مثال واحد في عمليات المشاهدة سابقاً. إن الأشياء المنظورة بيانياً التي تم تعريفها في فضاء WC يتم تحويلها إلى أشياء منظورة في فضاء NDC، ومن ثم تحول مرة ثانية إلى أشياء منظورة في فضاء DC.

تظهر التحويلات في عدة طرق أخرى. تتضمن الرسومات المتحركة نقل أشياء منظورة بيانياً في الوقت المناسب ويتم إنجاز هذا اعتيادياً بواسطة تحويل هذه الأشياء من مكان لآخر. التطبيقات كتصميم دائرة كهربائية أو وضع مخطط لغرفة تعتمد على مناورة ومعالجة عدد من أشياء منظورة (كالمقاومات ودوائر متكاملة ومناضد وكراسي) ووضعها في مواقع مطلوبة في مخطط التصميم. يتم تحقيق الموقع من خلال سلسلة من التحويلات. تمتاز كثير من معماريات منظومة الرسومات بالحاسوب على احتوائها سلسلة من مكونات مادية للتحويلات التي تعمل على كيانات أولية للرسومات حيث ينتقل الأخير من خلال المنظومة ابتداءً من تعريفه في برنامج تطبيقي إلى أن يتم عرضه أخيراً على جهاز إخراج.

ستكون خطوتنا الأولى هي تطوير الرياضيات الضرورية لصنف من تحويلات محافظة على الخط (Line- Preserving Transformation) تعرف بتحويلات تآلفية (Affine Transformation) وحالما ننتهي من هذا، سنقوم بتطوير حزمة برامج بسيطة التي تستخدم بالتضامن مع منظومة الرسومات.

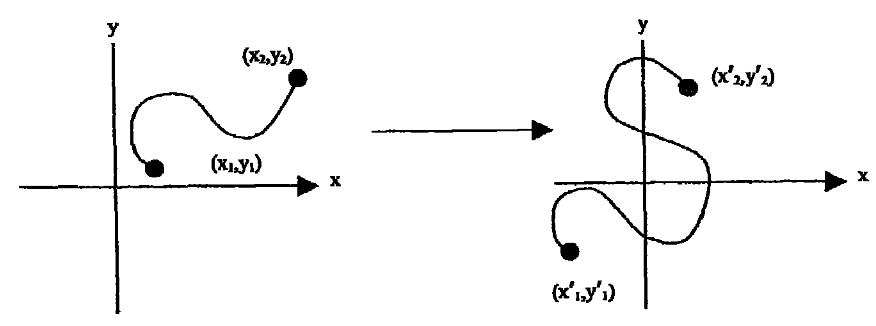
بعد ذلك ، ستتحول إلى استخدام التحويلات في النمذجة. حتى الآن، قمنا بتمثيل عالم التطبيق بطريقة خطية. تستخدم النماذج المتطورة العلاقات ما بين الأشياء المنظورة

لبناء نماذج من العالم الذي حولنا. على سبيل المثال، تكون دواليب السيارة متصلة منطقياً ومادياً بهيكل السيارة. سنقوم بدراسة طرق للتعبير عن هذه العلاقات ، وبعدئذ نستخدم النماذج الناتجة في منظومتنا للرسومات.

1.5 تحويلات تآلفية (افينية) (Affine Transformations)

1.1.5 تحويلات عامة (General Transformations)

في أغلب الأحيان، الكيانات الأولية التي نستخدمها في منظومة الرسومات تعتمد على الخطوط. هناك أسباب متعددة تؤدي إلى ظهور الخطوط بصورة طبيعية في عالمنا الحقيقي للتطبيقات، وسهولة توليدها، وكونها مناسبة لموضوع نقاشنا وسهولة تحويلها. لنأخذ بنظر الاعتبار مشكلة تحويل أو نقل قطعة منحنى C بين نقطتين (x_1, y_1) و (x_2,y_2) إلى قطعة منحنى جديدة C' بواسطة تحويل سنشير إليه بالرمز T كما هو مبين في الشكل 1.5.



الشكل 1.5 تحويلات عامة

باستخدام طريقة تمثيل المتجه، فإن أي نقطة $p = \begin{bmatrix} x \\ y \end{bmatrix}$

على المنحنى C يتم تحويله إلى نقطة جديدة على المنحنى C' حيث أن

$$p' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

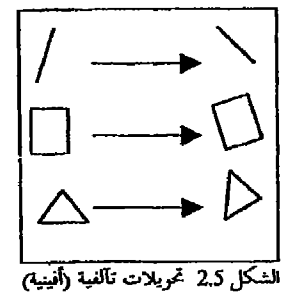
يمكننا التعبير عن هذه العلاقة بما يلي: $P' = T(p)$

حيث T الصيغة التي تعطي الوصف الطبيعي للتحويل بالضبط.

لنفترض نحن نرغب باستخدام منظومتنا للرسومات لعرض المنحنى الذي تم تحويلـــه. بصورة عامة، لا المنحنى C ولا المنحنى C' ســيطابق الكيانــات الأوليــة في منظومتنـــا للرسومات. قد نحاول بطريقة ما تشبه تحويل مجموعة النقاط $\{p_i\}$ الواقعـــة علـــى C إلى مجموعة نقاط $\{p_i'\}$ واقعة على C'. ومن أجل الحصول على صورة للمنحنى C'، يمكننـــا استخدام النقاط التي تم تحويلها لتكوين كيان أولي متعدد الخطوط الذي يقرب المنحـــنى C' ويمكن عرضه على معظم منظومات الرسومات. مع أن هذه الطريقة ممكنة، ولكنها قـــد تحتاج إلى عمليات حسابية هائلة لكي تكون طريقة نافعة في عرض المنحنيات المحولة.

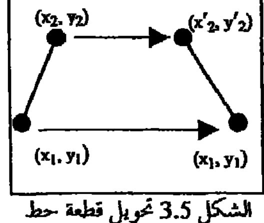
2.1.5 تحويل خطوط إلى خطوط (Transforming Lines To Lines)

كبديل، لنأخذ بنظر الاعتبار إحدى السمات الأكثر مقبولة للخط، حيث يتم تعريف الخط تماماً بواسطة أي نقطتين تقع عليه. إذن، نحتاج خزن هاتين النقطتين فقط، وفي حالـة قطعة الخط، خزن نقطتي النهاية القطعة. تكون هذه وســيلة فعالة للخزن، حيث يكون بوسعنا استخدام الكيان المـادي أو البرمجيات لتوليد كل النقاط الأخرى عند الحاجة، وعادة عنـد الإخراج إلى العارضة. لو تم تحويل خط إلى شيء آخر غـــير الخط، يعني هذا نحن لم نتوصل إلى حل مشكلتنا. لكن، مـــن حسن الحظ، كثير من العمليات المهمة جداً التي تؤديـــها في



الشكل 2.5 تحويلات تآلفية (أفينية)

عالم الحقيقة تحافظ على الخطوط. هذه التحويلات تشمل على التدوير والانعكاس وتغيـير الأبعاد والانتقال، جميعها تنتمي إلى صنف من التحويلات تعرف بـــالتحويلات التآلفيـــة (Affine Transformations) كما مبين في الشـــكل 2.5. إن أهمية التحويلات التآلفية في الرسومات بالحاسوب ينبغـــي أن تكون واضحة.



الشكل 3.5 تحويل قطعة خط

لنفترض لدينا قطعة خط تصل بين نقطتـــين (x_1, y_1) و (x_2, y_2). إذا قمنا بتحويل هاتين النقطتين بصورة انفراديـــة عن طريق التحويل التآلفي إلى نقطتين (x'_1, y'_1) و (x'_2, y'_2) على التوالي كما في الشـــكل

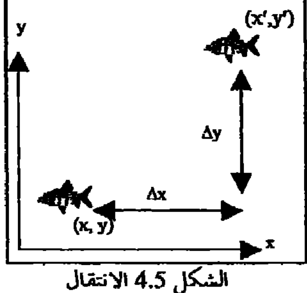
3.5. نحن نعلم أن جميع نقاط الوسط يتم توليدها بواسطة قطعة الخط الواصلة بين نقطتي النهاية الجديدة هذه . نظراً لكون التحويلات التآلفية تقوم بتحويل قطع خطوط إلى قطع خطوط، إذن أي تحويلين تآلفيين بالتعاقب يكون متكافئاً إلى تحويل تآلفي واحد. هذه الحقيقة سوف تتيح لنا بناء تحويلات مركبة وذلك بضم سلسلة من التحويلات البسيطة. ستكون التحويلات الثلاثة الأساسية المستخدمة هي:

1. التدوير حول نقطة الأصول (Rotation About Origin)،
2. الانتقال (Translation)،
3. تغيير الأبعاد (التقييس) (Scaling).

في جميع الحالات، ينبغي علينا أن نتذكر أن هنالك عدد غير منتهي من الطرق في تحريك نقطة p إلى نقطة جديدة p'. عندما نعتبر هذه النقطة هي نقطة منفردة على شيء منظور ونفس العملية يتم تطبيقها على جميع النقاط الواقعة على الشيء المنظور، لذا سنرى هناك عادة وصف واحد للتحويل فقط.

3.1.5 الانتقال (Translation)

الانتقال عبارة عن إزاحة جميع النقاط التي تقع على الشيء المنظور بمسافات متساوية، كما مبين في الشكل 4.5. بإمكاننا وصف هذه العملية بواسطة زوج من المعادلات:



الشكل 4.5 الانتقال

$$x' = x + \Delta x$$
$$y' = y + \Delta y$$

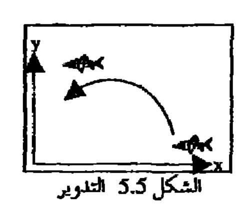
وبدلالة التمثيل بالمتجهات، فإذا كان

$$p = \begin{bmatrix} x \\ y \end{bmatrix} \quad , \; p' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad , t = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

فإنه بالإمكان كتابة عملية الانتقال كمجموع متجهات (أعمدة مصفوفة):

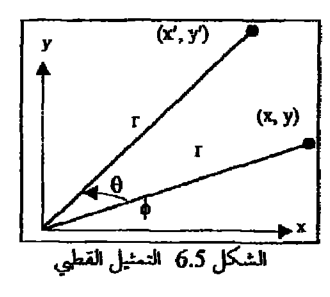
$$p' = p + t$$

4.1.5 التدوير (Rotation)



الشكل 5.5 التدوير

التدوير حول نقطة الأصل بزاوية 'θ' درجة موضح في الشكل 5.5، باستطاعتنا استخراج معادلة التدوير بملاحظة ما يلي، أثناء تدويرنا لنقطة حول نقطة الأصل تبقى تلك النقطة على بعد ثابت من نقطة الأصل. وباستخدام الصيغة القطبية (Polar Form) الشكل 6.5 نحصل على:



الشكل 6.5 التمثيل القطبي

$$x = r\cos\phi$$
$$y = r\sin\phi$$
$$x' = r\cos(\theta + \phi)$$
$$y' = r\sin(\theta + \phi)$$

وباستخدام الصيغ المثلثية (Trigonometric Formulae) للجيب والجيب تمام (sine and cosine) لمجموع زاويتين، نجد:

$$x' = r\cos\theta\, co\phi - r\sin\theta\sin\phi$$
$$= x\cos\theta - y\sin\theta$$
$$y' = r\cos\theta\sin\phi + r\sin\theta\cos\phi$$
$$= x\sin\theta + y\cos\theta$$

يمكن التعبير عن هذه المعادلات كحاصل ضرب بين مصفوفة ومتجه (matrix – vector multiplication):

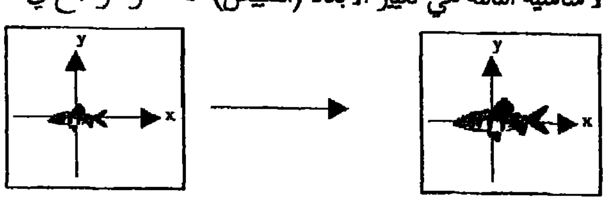
$$p' = Rp$$

حيث R هي المصفوفة التالية :

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

5.4.5 تغيير الأبعاد : (Scaling)

تكون العملية الأساسية الثالثة هي تغيير الأبعاد (التقييس) كما هو موضح في الشكل 5 .7.



الشكل 7.5 تغيير أبعاد

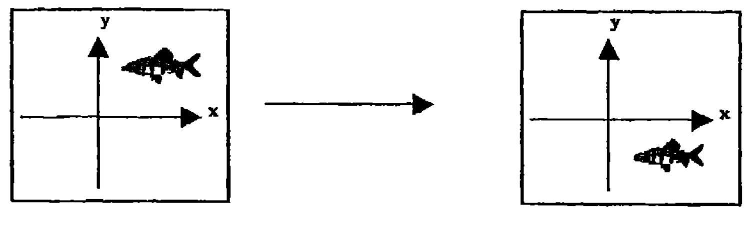
سنسمح بتغيير الأبعاد للاتجاهين y,x بصورة منفصلة . ثوابت تغيير الأبعاد α, β تحدد مقدار تغيير الأبعاد في كل اتجاه:

$$x' = \alpha x$$
$$y' = \beta y$$

إذا كان ثابت تغيير البعد أكبر من واحد، هذا سيؤدي إلى امتداد أبعاد الشيء المنظور بذلك الاتجاه. وإذا كان الثابت موجب ولكن اقل من واحد، يكون هناك انكماش على امتداد ذلك الاتجاه.

معامل تغيير الأبعاد السالب يؤدي إلى حدوث عملية انعكاس حول المحور المناظر. على سبيل المثال، في الشكل 8.5، عندنا $\alpha=1$ و $\beta=-1$. كذلك يمكن التعبير عن تغيير الأبعاد بواسطة عملية مصفوفة ومتجه : $p' = Sp$,

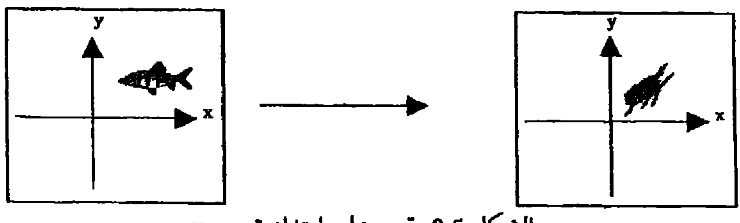
$$S = \begin{pmatrix} \alpha & o \\ o & \beta \end{pmatrix}$$



الشكل 8.5 الانعكاس

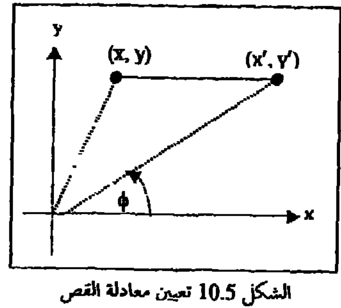
6.1.5 قص : (shear).

هناك عملية أخرى، تدعى القص، كما مبين في الشكل 9.5.



الشكل 9.5 قص على امتداد محور -x

أيضاً يكون له فائدة في تشكيل تحويلات تآلفية عامة. في هذا الشكل قد تم توضيح قص على امتداد المحور x. الزاوية ϕ تحدد مقدار القص. شكل مشابه قد يوضح القص على امتداد المحور y.



الشكل 10.5 تعيين معادلة القص

يمكن الحصول على معادلات القص باتجاه x ببساطة وذلك باستخدام قوانين المثلثات كما هو مبين في الشكل 10.5. تكون هذه المعادلات كما يلي:

$$x' = x + y \cot \phi$$

$$y' = y,$$

والتي تؤدي إلى مصفوفة القص في اتجاه المحور x:

$$H_x = \begin{bmatrix} 1 & \cot \phi \\ 0 & 1 \end{bmatrix}$$

نستطيع إيجاد مصفوفة القص للمحور -y بنفس الأسلوب. بالرغم من أن القص يمكن استخراجه من عمليات أخرى، ولكونه يستخدم في كثير من الأحيان لذا سنعتبره أحد العمليات الأساسية.

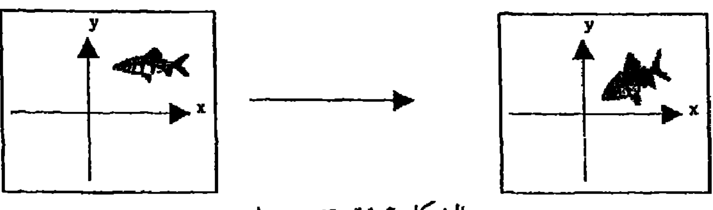
إن هذه العمليات الأساسية الأربعة ليست كافية لوصف جميع التحويلات التآلفية إذا تم أخذ كل واحدة بصورة انفرادية. على سبيل المثال، قد نرغب بتدوير شيء منظور حول نقطة غير نقطة الأصل، أو إجراء عملية انعكاس لشيء منظور حول محور اختياري (Arbitrary Axis). يمكننا تحقيق هذه العمليات بواسطة جمع هذه العمليات الأساسية.

2.5 سلسلة التحويلات (Concatenating Transformations)

تستطيع استخدام تحويلاتنا الأساسية لتوليد تحويلات أكثر شمولية محافظة على الخط. لو قمنا بتطبيق تحويلين تآلفيين بالتعاقب على قطعة خط، فإن مع ذلك سيكون لدينا قطعة خط. إذن، التحويل الناتج من عملية توحيد التحويلين أيضا يكون تحويل تآلفي. إن عملية التسلسل (Concatenation) للتحويلات هي إحدى العمليات الفعالة التي تؤدي للحصول على تحويلات أكثر شمولية مما لدينا الآن. لنبدأ مع مثال بسيط، سنجد الحاجة إلى إعادة تقييم كيفية تمثيل تحويلاتنا لكي نحصل على طريقة كفوءة لتركيب هذه التحويلات.

1.2.5 التدوير حول نقطة ثابتة (Rotating About a Fixed Point)

افترض نحن نأخذ بنظر الاعتبار تدوير حول نقطة ليست نقطة الأصل، (x_f, y_f) كما في الشكل 11.5.

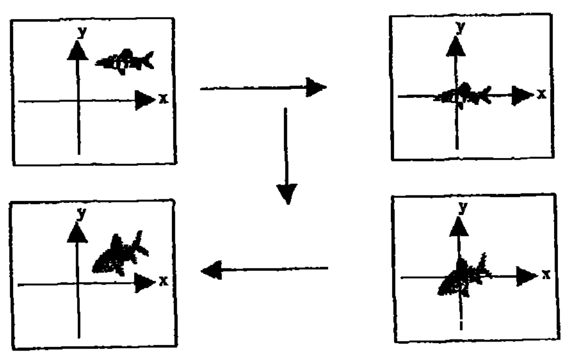


الشكل 11.5 تدوير عام

تدعى هذه النقطة بالنقطة الثابتة للعملية، نظراً لكونها لا تتغير من جراء التدوير. نستطيع استخراج المعادلات لهذا التدوير وذلك باستخدام مثلثات أولية كما فعلنا مع التدوير حول نقطة الأصل. كطريقة مشوقة أكثر هي باستخراج المعادلات مستخدمين العمليات الأساسية التي هي الآن ضمن ذخيرتنا.

يمكن حل هذه المشكلة في ثلاثة خطوات:

1- نحن نعلم كيف يتم التدوير حول نقطة الأصل، لذا كطريقة ممكنة (غالباً ما تكون طريقة جيدة) هي تحويل المشكلة لدينا إلى واحدة سبق وقمنا بحلها. نقوم بتحويل هذه المشكلة أولاً بنقل النقطة الثابتة إلى نقطة الأصل. تستلزم هذه الخطوة انتقال جميع النقاط بمقدار $(-x_f, -y_f)$، كما في الشكل 12.5.



الشكل 12.5 تحريك النقطة الثابتة

$$\bar{x} = x - x_f$$
$$\bar{y} = y - y_f$$

2- الآن يمكننا تدوير حول نقطة الأصل كما من قبل، فتنتج النقطة (x , y)

$$\hat{x} = \bar{x} \cos\theta - \bar{y} \sin\theta$$
$$\hat{y} = \bar{x} \sin\theta + \bar{y} \cos\theta$$

لا تزال العملية، لكوننا قمنا بإزاحة كل شيء بمقدار $(-x_f, -y_f)$. نستطيع إرجاع هذه الإزاحة وذلك ببساطة إجراء عملية النقل مرة ثانية بنفس المقدار لكن بالاتجاه المعاكس. وإعادة النقطة الثابتة إلى موقعها الأصلي. إذن، نقطتنا الأخيرة تكون (x', y')، حيث

$$x' = \hat{x} + x_f$$
$$y' = \hat{y} + y_f$$

هذه العملية الأخيرة هي معكوس عملية النقل الأولى. غالباً ما نستخدم الحقائق وهي، ليس فقط كل عملية من عسملياتنا الأساسية تملك معكوس الذي يقوم بإرجاع العملية، لكن يجب تكون العملية المعكوسة من نفس النوع. كما رأينا هنـاً، معكـوس الانتقال هو نفسه انتقال أيضاً. الآن، بواسطة ضم هذه المعادلات، نحصل على معادلات تدوير عامة:

$$x' = (x - x_f) \cos\theta - (y - y_f) \sin\theta + x_f$$
$$y' = (x - x_f) \sin\theta + (y - y_f) \cos\theta + y_f$$

2.2.5 احداثيات متجانسة (Homogeneous Coordinates)

عند هذه المرحلة ، قد نتسائل لماذا لا نقوم باستخدام صيغ التمثيل بالمصفوفات والمتجهات التي سبق تقديمها مع التحويلات الأساسية. كان بالإمكان استخدامهم، لكن نتائجها قد لا تكون مرضية بصورة خاصة. العملية الأولى، تتضمن حركة النقطة الثابتة إلى نقطة الأصل، حيث يمكن تمثيلها بواسطة إضافة متجهات. وأما العملية الثانية- التدوير- هي عملية ضرب مصفوفة-متجه. والعملية الأخيرة هي عبارة عن إضافة متجه آخر. الآن المشكلة هي كيف نقوم بضم هذه العمليات الثلاثة في صيغة واحدة. هنا يكون الحل صعب، نظراً لكوننا لا نستطيع ضم إضافة متجه وحاصل ضرب مصفوفة-متجه في

عملية واحدة مكافئة في البعد الثنائي. مع هذا، يمكننا تحقيق تمثيل دقيق لو انتقلنا إلى البعد الثلاثي.

توفر لنا الإحداثيات المتجانسة تمثيل خاص لثلاثي الأبعاد وتتيح لنا أيضاً معالجة سهلة للكيانات ثنائية الأبعاد. نقوم باستبدال نقطة ثنائية الأبعاد $p = \begin{bmatrix} x \\ y \end{bmatrix}$ بنقطة ثلاثية الأبعاد

$$p = \begin{bmatrix} \omega x \\ \omega y \\ \omega \end{bmatrix}$$

يكون التمهيد الوحيد هو على المتغير ω الذي ينبغي أن لا يكون صفراً، مع هذا التحديد يمكننا التنقل بين نقطة ما وتمثيلها بإحداثيات متجانسة وذلك بالضرب أو القسمة على ω حسب الضرورة. في الرسومات الاعتيادية ثنائية الأبعاد، دائماً يمكننا جعل قيمة ω مساوية لواحد ، وهنا سنفترض أننا اتخذنا مثل هذا الاختيار.

باستخدام إحداثيات متجانسة، يمكن تمثيل عملياتنا الأساسية الثلاثة بواسطة مصفوفات ، كما هو الحال مع تجميع هذه العمليات. على سبيل المثال، انتقال نقطة (x,y) الممثلة بإحداثيات متجانسة

$$p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

بمقدار $(\Delta x, \Delta y)$ إلى موقع جديد (x', y') الذي يمكن التعبير عنه كما يلي:

$$p` = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = T(\Delta x, \Delta y)\, p,$$

حيث أن $T(\Delta x, \Delta y)$ هي المصفوفة التالية:

$$T(\Delta x, \Delta y) = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix}$$

وبإجراء عملية ضرب المصفوفة- المتجه، يمكننا بسهولة البرهنة على إمكانية الحصول على معادلات الانتقال الأصلية من أول صفين للمصفوفة. يبقي الصف الثالث المقدار 'ω'

بدون تغيير بسبب عملية الانتقال، وهكذا، هذه المصفوفة ستبقى نفسها حتى لو سمحنـــا لقيم ω أن تكون غير الواحد.

لو اتبعنا نفس النهج بالنسبة للتدوير حول نقطة الأصل، وتغيير الأبعاد، والقص علـى امتداد المحور -x، سنحصل على المصفوفات الثلاثة التالية:

$$R(\theta) \quad \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & \end{pmatrix}$$

$$S(\alpha,\beta) = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$H_x(\phi) = \begin{pmatrix} 1 & \cot\phi & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3.2.5 تمثيلات مصفوفية (Matrix Representations)

الآن،يمكننا أن نرى كيف نستطيع سلسلة التحويـــلات التآلفيـــة (Concatenate Affine Transformation) ببساطة. لنفترض قمنا بسلسلة من التحويلات مع المصفوفات C,B,A على نقطة P، حيث كل مصفوفة تمثل إحدى عمليتنا الأساسية. إذن، ســـتكون النقطة المحولة الناتجة هي: $p' = C(B(Ap))$

باستخدام الخواص الأساسية للمصفوفات، نستطيع إعادة كتابة هذه المعادلة كما يلي:

$$p' = Mp$$

حيث أن M تمثل المصفوفة التالية:

$$M = CBA$$

هكذا، يمكن تمثيل تحويلات تآلفية متتابعة بواسطة مصفوفة واحدة. لاحظ ذلك، بمـا أن، وبشكل عام

$$AB \neq BA$$

ينبغي علينا أن ننتبه حول ترتيب المصفوفات أثناء تكوين تحويلات مركبة. هنـــالك تحذير إضافي آخر هو أن يكون تطبيق ترتيب العمليات في CBAp بعكس اتجـــاه القـــراءة

الطبيعية (من اليسار إلى اليمين) في الانكليزية. عديد من كتب الرسومات والمنظومات تقوم بتبديل المتجهات العمودية بمتجهات صفية وذلك باستخدام عملية منقول المصفوفات (Matrix Transposition). تقوم عملية المنقول هذه بتحويل معادلتنا إلى صيغة مكافئة (حيث يشير الرمز العلوي T لمنقول المصفوفة) مع متجهات صفية هي :

$$P'^T = p^T A^T B^T C^T = P^T M^T,$$

التي يتم قراءها بالترتيب العكسي، مع ذلك، انسجاماً مع منظومة الرسومات الحديثة وأدبيات الفيزياء والرياضيات، سوف نستخدم صيغة العمود.

نعود إلى مثالنا للتدوير حول نقطة ثابتة، تكون خطواتنا الثلاثة هي:

$$A = T(-x_f, -y_f)$$

$$B = R(\theta)$$

$$C = T(x_f, y_f).$$

هذه تنتج المصفوفة المتسلسلة التالية:

$$M = CBA = \begin{pmatrix} \cos\theta & -\sin\theta & y_f\sin\theta - x_f\cos\theta + x_f \\ \sin\theta & \cos\theta & -x_f\sin\theta - y_f\cos\theta + y_f \\ 0 & 0 & 1 \end{pmatrix}$$

4.2.5 معكوس التحويلات (Inverse Transformations)

قبل البدء بشرح كيف يمكننا استخدام التحويلات مع منظومتنا للرسومات، دعنا نأخذ بنظر الاعتبار مزيداً من الأمثلة الأخرى. إحدى الصفوف المهمة من العمليات هي معكوس التحويلات. لنضرب مثال، لو استخدمنا تحويل تآلفي A لإنتاج نقطة محولة:

$$p' = A(p)$$

معكوس العملية ونرمز له بواسطة A^{-1}، حيث نحصل :

$$p = A^{-1}(p')$$

وباستخدام إحداثيات متجانسة، المصفوفة A تصبح مصفوفة 3x3 ، لذا

$$p' = Ap,$$

$$p = A^{-1}p'$$

حيث أن A^{-1} هو معكوس المصفوفةA. بالنسبة للعمليات الأساسية، هذه المعكوسات دائماً تكون موجودة، إلا إذا فعلنا شيئاً تافهاً كاستخدام معامل تغيير الأبعاد صفر (Scale Factor '0') وهكذا، المعكوسات تكون موجودة لجميع عملياتنا المركبة، لأن

$$(AB)^{-1} = B^{-1} A^{-1}$$

أما بالنسبة لعملياتنا الأساسية، تكون معكوس التحويلات واضحة جداً إذا قمنا ببساطة تطبيق أفكار هندسية. على سبيل المثال، معكوس التدوير بزاوية هو تدويرθ بزاوية θ-. هكذا بالنسبة للعمليات الأساسية الانتقال، التدوير وتغيير الأبعاد يكون معكوس التحويلات كالتالي:

$$T^{-1}(\Delta x, \Delta y) = T(-\Delta x, -\Delta y),$$

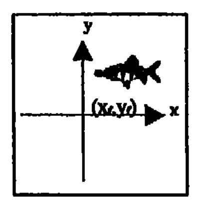
$$R^{-1}(\theta) = R(-\theta),$$

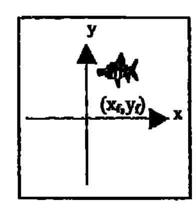
$$S^{-1}(\alpha,\beta) = S\left(\frac{1}{\alpha},\frac{1}{\beta}\right),$$

$$H_x^{-1}(\phi) = H_x(-\phi).$$

5.2.5 أمثلة على السلسلة (Concatenation Examples)

الآن ، كثير من عملياتنا الأكثر تفصيل يمكن التعبير عنها بدلالة العمليات الأساسية ومعكوسها. على سبيل المثال، لنأخذ بنظر الاعتبار تغيير الأبعاد حول نقطة ثابتة (x_f,y_f) كما مبين في الشكل 13.5.





الشكل 13.5 تغيير الأبعاد بصورة عامة

وكما هو الحال مع التدوير، تبقى النقطة الثابتة بدون تغيير عبر التحويل. إن العملية الأساسية لتغيير الأبعاد تكون لها نقطة ثابتة للأصل . نقوم بتبني طريقة مشابهة لتلك التي استخدمت في عملية التدوير العامة وهي:

1. نقل النقطة الثابتة إلى نقطة الأصل.

2. تغيير الأبعاد
3. ومن ثم تحريك النقطة الثابتة والعودة إلى موقعها الأصلي.

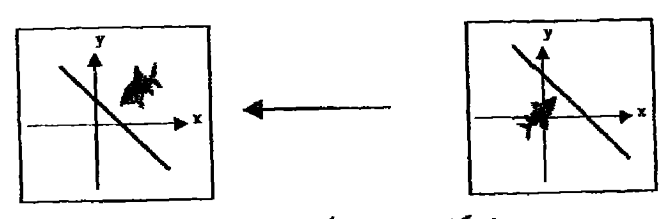
هكذا تكون المصفوفة 3x3 المطلوبة هي

$$A = T(x_f, y_f)\, S(\alpha, \beta)\, T(-x_f, -y_f)$$

$$= \begin{pmatrix} \alpha & 0 & (1-\alpha)\, x_f \\ 0 & \beta & (1-\beta)\, y_f \\ 0 & 0 & 1 \end{pmatrix}$$

أيضاً يمكننا ملاحظة إن الصفوف السفلى في كل مصفوفة تحويل تكـــون متماثلـــة. تكون هذه الصفوف متماثلة لأن وجودها هناك مجرد من أجل إبقاء المتغـــير الإضـــافيω بدون تغيير. بالنسبة لتطبيقات ثنائية البعد نحتاج إلى إبقاء أول صفين فقـــط أو بشـــكل مكافئ لمصفوفة 2x3.

كمثال أخير، لنأخذ بنظر الاعتبار الانعكاس (Reflection) لشيء منظور حول خط اختياري (Arbitrary Line)، كما هو موضح في الشكل 14.5.



الشكل 14.5 الانعكاس حول محور

نحن نعلم بالإمكان إيجاد معكوس شيء منظور حول المحور -y وذلـــك باســـتخدام مصفوفة تغيير الأبعاد:

$$S(-1,1) = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

مرة ثانية، سوف تحول هذه المشكلة إلى مشكلة يكون حلها معروف. افترض محـــور الانعكاس هو عبارة عن خط معادلة هي:

$$y= mx+h$$

ابتدأ مع الميل m، يمكننا إيجاد الزاوية

$$\theta = \tan^{-1} m$$

التي يصنعها الخط مع محور -x. نستطيع ترصيف (Align) محور التدوير مع محـور -y وذلك بتدويره زاوية مقداره ϕ ($\phi = 90 - \theta$) درجة. لاحظ ذلك، يكون اختيـار جيـب وجيب تمام الزاوية كالتالي:

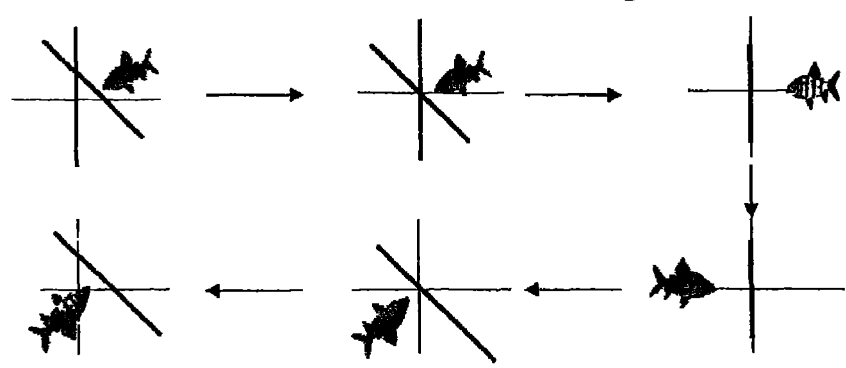
$$\sin\phi = \frac{1}{\sqrt{1+m^2}}$$

$$\cos\phi = \frac{m}{\sqrt{1+m^2}}$$

كذلك علينا إزاحة المحور المدار إلى نقطة الأصل مع T (o,-h). إذن هنالك خمسـة تحويلات أساسية نستخدمها لتركيب التحويل المطلوب وهي:

1. الإزاحة إلى نقطة الأصل (Shift To The Origin).
2. تدوير لترصيف المحور (Rotation to Align the Axis)
3. الانعكاس حول المحور -y (Reflection About The y-Axis)
4. تدوير إلى الوراء (Rotation Back)
5. نقل إلى الوراء (Translation Back)

كما مبين في الشكل 15.5.



الشكل 15.5 سلسلة من التحويلات

هكذا تكون المصفوفة المركبة :

$$A = T(0,h)\,R(-\phi)\,S(-1,1)\,R(\phi)\,T(0,-h).$$

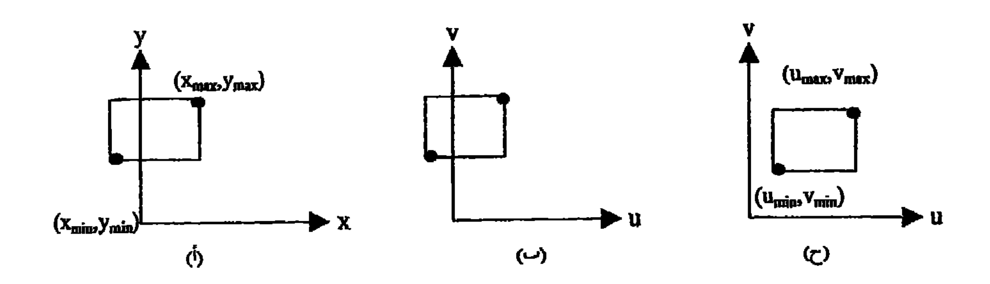
وبنفس الأسلوب، يمكننا بناء تحويلات تآلفية اختيارية. توجد هنالك عدة طرق تمكننا من تركيب تحويل معين. وبغض النظر عن الطريقة المتبعة لاستخراج التحويل، ينبغي أن تكون المصفوفة المتسلسلة الناتجة هي نفسها. أيضا من المهم ملاحظة ذلك، نظراً لكوننا نستخدم غالباً تحويل تآلفي معين في تحويل مئات بل ألوف من النقاط (أو حتى ألوف من الأشياء المنظورة)، يكون العمل الإضافي في استخراج التحويل من مجموعة تحويلات بسيطة يمكن إهماله. الآن نستطيع العودة إلى معرفة كيفية استخدام التحويلات في منظومات الرسومات.

3.5 التحويلات في منظومة GKS (Transformations in GKS)

من أجل فهم استخدام التحويلات في منظومة GKS وفي عديد من منظومات الرسومات الأخرى، علينا العودة إلى دراستنا السابقة المتعلقة بمنظومات الإحداثيات والمقارنة مع آلة التصوير الاصطناعية، يتم تعريف الكيانات الأولية في فضاء إحداثيات WC ومن ثم تحويلها إلى فضاء إحداثيات ADC مستخدماً شروط المشاهدة الحالية. عندما نرغب بتطبيق تحويل ما، يمكننا تطبيقه أما على كيان أولي في فضاء WC (يعني هذا، قبل تحويله إلى فضاء NDC) أو يمكننا تطبيقه بعد التحويل المعياري (Normalization Transformation). هاتين الطريقتين تعطي نتائج مختلفة جداً.

إحدى الطرق لمعرفة سبب تباين النتائج هي العودة في تفحص معادلات التحويل من WC إلى NDC التي تم استعراضها في الفصل الثاني. يمكننا تمثيل هذا التحويل المحافظ للخط وذلك باستخدام تحويلين من عملياتنا الأساسية هما: الانتقال وتغيير الأبعاد. وهذا يتكون من :

1- تغيير الأبعاد في WC كما مبين في الشكل 16.5، نقطة ثابتة تقع في مركز النافذة، إلى حجم بوابة الرؤية، يتبعه التحويل التالي:



الشكل 16.5 التحويل المعياري

أ - إحداثيات كونية. ب- تغيير الأبعاد ج-الانتقال

2- انتقال الزاوية السفلى لليسار للنافذة إلى الزاوية السفلى لليسار لبوابة الرؤية (في التمرين 1.5 سنطلب منك برهان ذلك، وهو الحصول على نفس المعادلات كما في الفصل الثاني). لنفرض أننا أشرنا إلى المصفوفة التي تعرف هذا التحويل في إحداثيات متجانسة بـ N. الآن لنأخذ بنظر الاعتبار تحويل تآلفي معين تم وصفه بواسطة المصفوفة A حيث نرغب تطبيقه في فضاء WC. ما سنجده في فضاء NDC هي الكيانات الأولية الأصلية المعرفة في WC وتم تحويلها بواسطة المصفوفة المركبة التالية:

$$B = NA$$

والتي يتم تطبيقها على كل نقطة $P' = Bp$

وأما من ناحية الأخرى، لو قمنا بتطبيق A في فضاء NDC نحن فعلياً استخدمنا تحويل مركب كالآتي:

$$C = AN$$

نظراً لكون المصفوفات بصورة عامة غير إبدالية (Not Commute) ، تكون نتائج تطبيق B و C على كيان أولي مختلفين جداً.

باستطاعتنا التمييز بين هاتين الطريقتين في تطبيقاتنا للتحويلات وذلك بالإشارة إلى تطبيق التحويل على الأشياء المنظورة أولاً كتحويل شيء منظر أو نمذجي (Object Or Modeling Transformation). أما الحالة التي يتم فيها تطبيق التحويل بعد عملية الرؤية

أو المشاهدة تدعى تحويل الصورة (Image Transformation). تقـــوم منظومـــة GKS والمنظومات الأخرى كمنظومة CORE بتزويد تحويلات الصورة وتترك تحويلات الشـــيء المنظور إلى برنامج التطبيق.

تعطي منظومة GKS للمستفيد إمكانية تحويل الصورة من خلال تحويلات القطـــع. تعتبر مصفوفة تحويل 2x3 بمثابة صفة لكل قطعة. كما لاحظنا سابقاً، نحتاج تحديـــد أول صفين من المصفوفة 3x3 التي تعرف تحويل تآلفي في إحداثيات متجانسة. في بادئ الأمـــر، يتم إعداد هذه المصفوفة لتكون متساوية إلى أول صفين من المصفوفة المحـــايدة (Identity Matrix):

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

يمكن تغيير هذه المصفوفة بواسطة دالة GKS "ثبت تحويل القطعـة " (set segment tranformation)

```
void gset – seg-tran (seg – name, tran – matrix)
Gint seg – name;
Gfloat tran – matrix [2] [3];
```

يتم تثبيت المصفوفة A لتكون مساوية إلى tran – matrix ومن ثم تطبق على كيانات أولية في القطعة seg-name بعد أن يتم تحويلها إلى فضاء NDC أولاً. أيضاً توفر معظـــم منظومات الرسومات دالتين مساعدة (Two Utility Functions) هما:

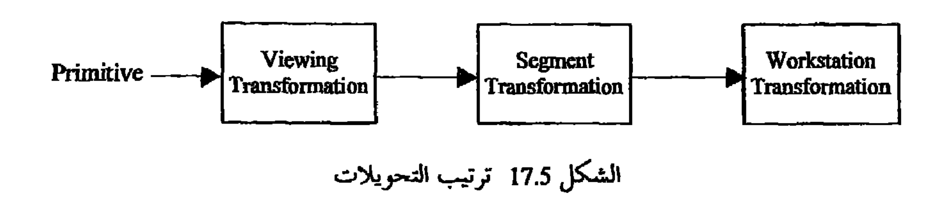
1- تقييم مصفوفة التحويل (Evaluate Transformation Matrix).

2- تراكم مصفوفة التحويل (Accumulate Transformation Matrix)

التي تساعدنا في تثبيت أو تغيير مصفوفة التحويل tran-matrix. الآن الأكثر أهميـــة هو لنرى ماذا يمكننا عمله وماذا لا يمكن عمله مع هذه الوسائل.

في كثير من الأحيان ، طلاب الرسومات بالحاسوب يواجهون صعوبات كبيرة مـــن أجل معرفة الفرق بين تحويلات الشيء المنظور وتحويلات الصورة وسبب توفير منظومـــة GKS تحويلات الصورة فقط ، لنتذكر ذلك، في منظومة GKS، تكون القطع مرافقة مـــع

محطات العمل. ومن الناحية المفاهمية، كل محطة عمل التي كانت فعالة أثناء تكوين القطعة لديها نسخ من القطع الخاصة بها. لذا يمكن وضع القطعة على محطات العمل هذه، ويتم أداء عملية الرؤية على تلك الكيانات الأولية الموجودة في القطعة. إذن يتم إجراء تحويل القطعة بعد أن تكون موجودة على محطة العمل وبعدئذ يتم تحويل محطة العمل، كما بين في الشكل 17.5.



الشكل 17.5 ترتيب التحويلات

إن هذا الاختيار يتيح لنا التنفيذ الفعلي "لمحطة العمل المنطقية" لكي تشمل مكونات مادية لإجراء تحويلات القطع بصورة سريعة.

إحدى الوظائف التي يمكن تحقيقها بشكل متقن هو استخدام تحويلات قطعة في تزويم (التكبير عن قرب) أو تحريك الصورة (Panning). على سبيل المثال، المصفوفة التالية:

$$A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}$$

تضاعف حجم الصورة في القطعة، بينما المصفوفة

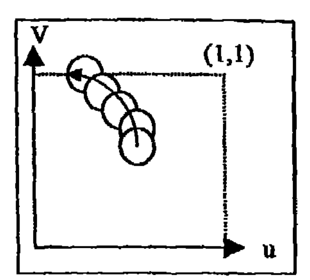
$$A = \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & -0.5 \end{bmatrix}$$

تقوم بإزاحة صورة القطعة للأسفل وإلى اليمين. كل هذا يبدوا حسناً وجيداً، ولكن عندما نتفحص مشكلة التدوير نواجه صعوبات.

في فضاء NDC، تكون نقطة الأصل في الزاوية السفلى لليسار من الشاشة الفعلية. أن تطبيق مصفوفات في الصيغة التالية (في فضاء NDC):

$$A = \begin{pmatrix} \text{Cos}\,\theta & -\sin\theta & 0 \\ \text{Sin}\theta & \cos\theta & 0 \end{pmatrix}$$

سيؤدي إلى التدوير المبين في الشكل 18.5 ، بدلاً من التدوير الاعتيـــادي المطلـــوب للشيء المتطور حول مركزه. عند أول نظرة، قد يبـــدوا سهلاً حل هذه المشكلة. نستطيع استخدام أساليب البند السابق لتدوير الشيء المنظور الموجود في القطعة حـــول مركزه وذلك باستخدام المركز كنقطة ثابتة. مع ذلـــك، توجد هنالك مشكلة رئيسية.



الشكل 18.5 التدوير في فضاء NDC

نظراً لكون التحويل تم تطبيقه في فضـــاء NDC، تكون النقطة الثابتة أيضاً في فضاء NDC بقدر ما يتعلـق الأمر بهذا التحويل. إن المبرمج التطبيقي لا يمتلك طريقة بسيطة في معرفة أين تكون هـــذه النقطة بدون القيام بالتحويل من WC إلى NDC بنفسه. أي بمعنى آخـــر، ينبغـــي علـــى المبرمج تعقب نتائج عملية الرؤية في برنامجه، الذي (مع ذلك ممكن) ينتزع أحد الفوائـــد الرئيسية لمنظومة الرسومات الحديثة. تعتبر هذه النقطة إحدى النقاط الصعبة، وقد تســتفيد من هذا وذلك ببناء أمثلة بسيطة تتمكن من تنفيذها باليد.

ليست قدرات تحويل الصورة في منظومة GKS وفي منظومات عديدة أخرى هـــي عديمة الفائدة. مع ذلك أنها معدة لاستخدامات معينة وليست معدة لاستخدامات أخـرى. لو رغبنا في تزويم أو تحريك أو بناء عرض مركب من الصور ، ستكون تحويلات القطع في منظومة GKS ملائمة بصورة جيدة . من الناحية الأخرى ، التطبيقات التي تطلـــب مـــن المستفيد بناء ومعالجة أشياء منظورة في فضاء WC، التي سنقوم بتطوير قدرات التحويـــل الخاصة بها.

4.5 حزمة برامج التحويل *(A Transformation Package)*

لو بقينا مع منظومة GKS (أو معظم منظومات البرمجة القياسية الأخـــرى)، علينـــا توفير إجراءات خاصة بنا لتحويل الأشياء المنظورة. تكون الروتينات جميعها سهلة بعـــض الشيء. نحن بحاجة إلى عدد قليل من الإجراءات الأساسية وتكون المصفوفـــات صغـــيرة. سنعمل مع مصفوفات تكون سعتها 3x3. مع أن الإجـــراءات في بعـــض المنظومـــات،

كمنظومة GKS يستخدم الصفين في الأعلى فقط، لكننا نرغب أن نكون في توافق في ما سنفعله مع الرسومات ثلاثية الأبعاد، حيث تكون المصفوفات المربعة ضرورية.

ستزيد من إمكانية تنقلية برامجنا لو قمنا بإضافة المصفوفة كنوع جديد من البيانات:

```
typedef float Matrix [3][3];
```

يمكننا التعويض عن float بـــ Gfloat لو عملنا في أجواء منظومة GKS.

1.4.5 إجراءات الاحتساب (Evaluation Procedures)

يستطيع البدء ببناء حزمة برامج تحويل خاصة بنا مع التحويلات التآلفية الثلاثة الأساسية وهي: التدوير، الانتقال وتغيير الأبعاد. إن أول ثلاثة إجراءات ستقوم بتكوين أو احتساب المصفوفات التي تم الإشارة إليها بـــ $R(\theta)$, $T(\Delta x, \Delta y)$, $S(\alpha)$.

عبارات هذه الإجراءات هي كما يلي:

```
void ev- rotate (theta, result)
/* Evaluate a Rotation Matrix
for theta degrees about the origin
Note sine and cosine routines require
input in radians */
#define  DEG-TO-RAD 0.01745
float theta;
Matrix result;
{
      double cos ( ), sin (  );
      result [0] [0] = result [1][1] = cos (DEG-TO-RAD*theta);
      result [1][0] = sin (DEG – TO – RAD *theta);
      result [2][2]= 1;
      result [0] [1] = -result [1][0];
      result [2][0] = result [2][1]= result [0][2] = result [1][2]= 0.0;
}
void ev – trans (dx, dy, result )
```

```
     /* Evaluates a translation Matrix
     for translation by (dx, dy)*/
     float dx, dy;
     Matrix result;
{
     result [0] [0] = result [1] [1]=result [2] [2] = 1.0;
     result [0] [1] = result [1] [0] = result [2] [0] = result [2] [1] = 0.0;
     result [0] [2] = dx;
     result [1] [2]= dy;
}
void ev- scale (sx, sy, result)
/* evaluate a scaling matrix with a
fixed point at the origin and scale
factors sx and sy */

float sx, sy;
Matrix result;
{
      result [0] [1] =result [1] [0] = result [2] [0]
      = result [2] [1] = result [0] [2]= result [1] [2] = result [1] [2]=0.0;
      result [0] [0] =sx;
      result [1] [1] = sy;
      result [2] [2] = 1.0;
}
```

2.4.5 إجراءات تراكم (Accumulation Procedures)

كذلك نحتاج إلى إجراءات لضم هذه العمليات لتكوين عمليات أكثر تعقيداً من التي سبق تقديمها. إحدى الطرق لإجراء هذا الضم هو توفير إجراء عملية ضرب المصفوفة، لكي نقوم بتكوين مصفوفات من النوع AB. هذا الإجراء للمصفوفات 3x3:

```
void ac- matrix (matrix – a,matrix – b, result)
/*forms the matrix product
```

```
result = matrix – a * matrix – b*/

Matrix matrix – a, matrix – b, result;
{
    int i,j,k;
    Matrix temp;
    for (i=o ; i< 3; i++ ) for (j=o; j < 3 ; j++)
      {
          temp [i] [j]= 0.0;
          for (k =0; k<3; k ++) temp [i] [j]+=
          matrix – a [i] [k] * matrix – b [k][j];
      }
    for (i=0; i<3; i ++) for (j=0;j<3;j++)
    result  [i] [j]= temp [i][j];
}
```

في هذا الإجراء، لقد تم خلق مصفوفة مؤقتة temp، لكي نتمكن السماح بإعادة حاصل ضرب المصفوفتين ووضعه في إحداهما بعد إنهاء العملية الحسابية.

هذه الإجراءات الأربع ستتيح لنا خلق جميع التحويلات التآلفية. ومن المناسب أيضاً، يمكننا إضافة إجراء القص (shear). ومن الطبيعي، نستطيع دائماً خلق مصفوفة مباشرة بواسطة تثبيت جميع عناصرها بكل بساطة. في كثير من الأحيان، قد نرغب ضرب مصفوفة بإحدى المصفوفات الأساسية. يمكننا استخدام الإجراءات السابقة لتكوين مثل هذه الإجراءات الجديدة. على سبيل المثال، إجراء تراكم مصفوفة التدوير (Accumulate Rotation Matrix) يأخذ مصفوفة A ونقوم بتشكيل المصفوفة RA، حيث R هي مصفوفة تدوير البرنامج بلغة C يكون كما يلي:-

```
void ac-rotate (theta, m)
float theta;
Matrix m;
{
    matrix temp;
```

```
        ev-rotate (theta, temp);
        ac-matrix (temp,m,m);
    }
```

نستطيع تعريف الإجرائين ac-trans و ac-scale بنفس الطريقة.

3.4.5 تطبيق التحويلات (Applying the Transformations)

لنعود إلى المثال السابق، فنحن نستطيع تكوين مصفوفة التدوير حول نقطـــة ثابتـــة (x,y) بواسطة:

```
ev-trans (-x, -y, matrix );
ac-rotate (theta, matrix);
ac-trans (x,y, matrix);
```

يمكننا تناول الانعكاس (reflection) حول خط y=mx+h بواسطة:

```
ev-trans (-h, 0.0, matrix );
theta = atan(m);
ac- rotate (theta, matrix);
ac- scale (-1., 1., matrix);
ac-rotate (-theta, matrix);
ac- trans (h,0.0, matrix);
```

نحن بحاجة إلى إجراء أخير قبل البدء باستخدام هذه المجموعة من روتينـــات ضـــرب المصفوفات، في الواقع تحتاج إلى إجراء يقوم بتحويل نقطة بواسطة أحد هذه المصفوفـــات التي تم تشكيلها. أي بمعنى آخر، نحتاج إلى إجراء لإيجاد

$$p' = Ap.$$

سندعوا هذه الدالة باسم دالة تحويل النقطة (transform-point)

```
void transform –point (trans – matrix, old – point, new – point)
/*computes a new  point using homogenous coordinate transformation matrix */
Gpt *old – point , *new-point;
Matrix trans-matrix;
{
     new – point → x = matrix [0] [2] + matrix [0] [0] *old-point → x +
```

```
        matrix [0] [1] *old -point → y;
        new- point →y = matrix [1] [2] + matrix [1] [0] *old-point → x +
        matrix [1] [1] *old – point → y;
    }
```

لاحظ ذلك، في هذه الدالة، قمنا بافتراض أن المركبة الثالثة (ω) لكل نقطة تبقى بدون تغيير عبر التحويل. إذا تم تغيير الصف الثالث للمصفوفة matrix، لكان من الضروري تبديل النقطة الجديدة new-point وذلك بقسمة كلا المركبتين على القيمة المناسبة ω.

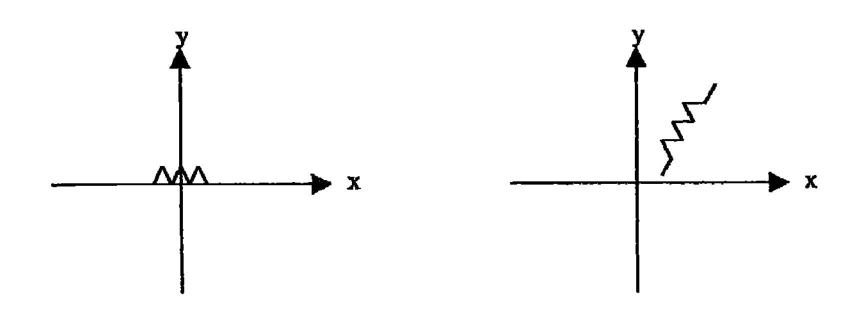
لنفترض لدينا شيء منظور، كالمقاومة في تطبيق تصميم دائرة كهربائية، التي تم وصفها بواسطة مصفوفة data لنقاط تعريف متعدد خطوط. وافترض أن هذا الشيء المنظور تم تعريفه حيث أن مركزه في نقطة الأصل. يمكننا تحويله إلى مواصفاتنا بواسطة:

1- تغيير أبعاده إما تكبيره أو تصغيره حسب الطلب،

2- تدويره،

3- تحريك مركزه.

هذه العمليات تنتج التحويلات المبينة في الشكل 19.5.



الشكل 19.5 تحويل شيء منظور

عبارات البرمجة الضرورية كما يلي، بافتراض أننا نرغب بوضع النقاط المحولة في متعدد خطوط آخر:

```
ev-scale (sx, sy, matrix );
ac-rotate (theta, matrix);
```

```
ac-trans (x,y, matrix);
for (i=0; i< npoints; (i++)
transform-point (matrix, &data [i] , &new data [i]);
gpolyline (npoints, newdata);
```

هذا السياق هو أحد الطرق المألوفة في تطبيقات النمذجة. سنقوم بدراسته بالتفصيل في البند القادم.

5.5 رموز وحالات مشاهدة (Symbols and Instances)

الآن نستطيع أن ننظر إلى طريقة استخدام التحويلات كأداة للنمذجة. أن المثال في البند السابق يعطينا تلميح حول ماذا نستطيع فعله. بدلاً من تعريف شيء منظور وأين نرغب أن يكون في عالم تطبيقاتنا، يمكننا تعريفه بأي أسلوب مناسب ومن ثم تحويله إلى الموقع المطلوب مع الاتجاه المطلوب ومعاملات تغيير الأبعاد. هنالك توجد عدد من الفوائد لمثل هذه الطريقة. غالباً، ما يكون من السهل تعريف أشيائنا بطريقة مبسطة وبعدها يبتدأ اهتمامنا إلى أين ستنقل. في عملية الرسوم المتحركة، قد يتم تحريك الشيء المنظور عدة مرات. من السهولة عادة أن يتم تعريفه مرة واحدة ومن ثم يستخدم تحويل ما لإعادة توليدة أثناء حركته. في كثير من التطبيقات، يتم استخدام عدة أشياء منظورة بصورة متكررة. على سبيل المثال، في تطبيق تصميم دائرة كهربائية بعض الأشكال الأساسية ، كالمقاومة والدوائر المتكاملة تظهر بصورة متكررة، في النموذج. في مثل هذه الحالات، عادة يكون من الأفضل تعريف كل شيء مرة واحدة واستخدام قدرات التحويل في وضع نسخ من النموذج المطلوب .

حتى الآن، أصبحت لدينا طريقة واحدة فقط في تكوين مجموعة من كيانات أولية للرسومات وبالتحديد من خلال قطعة (Segment). بقليل من التفكير سيتبين لنا أن هذه الطريقة ليست مثالية في معظم التطبيقات. على سبيل المثال، لنأخذ بنظر الاعتبار تصميم دائرة كهربائية مستخدمين مركبات منفصلة كالمقاومات والمكثفات ودوائر متكاملة. قد يحتوي التصميم الكامل على مئات من هذه المركبات. مع أنه، يمكننا رسم كل واحد من هذه المركبات (الأسلاك الواصلة بينهما) مستخدمين متعددات خطوط، ولكن لو تم تعريف كل واحد منها كقطعة لربما قد لا يكون لها معنى. قد يكون الاستخدام الأفضل

للقطعة لبعض مجموعة العناصر المرتبطة منطقياً في الدائرة الكهربائية، مثل تلك التي تشكل دائرة تضخيم (Amplifier) أو بعض الدوائر الفرعية الأخرى. مع ذلك، لا زلنا نود أن تكون لدينا طريقة لتمييز العناصر، كالمقاومات التي تكون أكثر تعقيداً من كياناتنا الأولية القياسية، كمتعددات الخطوط. أننا نستخدم المصطلح رمز (Symbol) للإشارة إلى مثل هذه الأشياء المنظورة.

1.5.5 الرموز (Symbols)

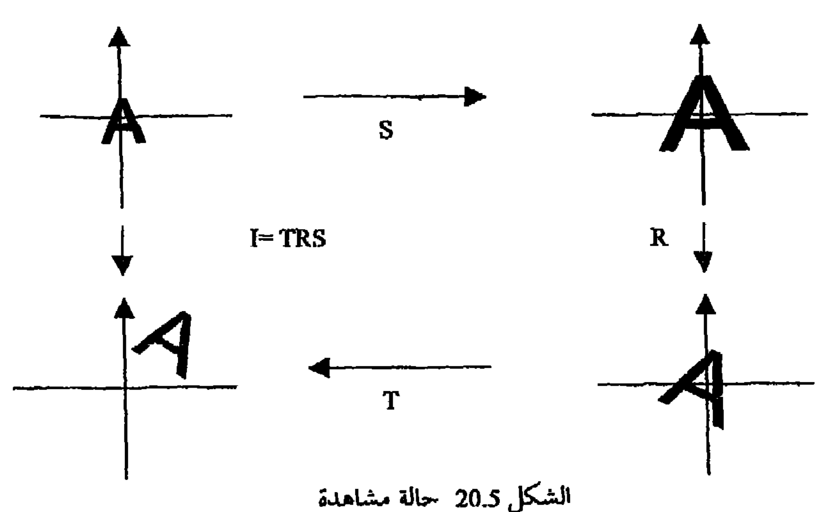
تعتمد الرموز على التطبيقات وتكون معرفة من قبل المستفيد. نستطيع اعتبار الرموز وكأنها معرفة بواسطة دوال، مثلاً ( ) resistor أو ( ) chair. في كل دالة، ستكون هنالك العبارات البرمجية الضرورية لتوليد الرموز مستخدمة الكيانات الأولية المتوفرة في منظومة الرسومات. على سبيل المثال، بالإمكان تعريف رمز المقاومة مستخدماً متعدد خطوط واحد. ولكن، الرموز لوحدها ليست كافية لإعطائنا المرونة المطلوبة في تصميم تطبيقاتنا. قد يظهر رمز معين عدة مرات في تطبيق. كل ظهور للرمز يعرف كحالة ظهور أو مشاهدة للرمز (Instance of the Symbol). من أجل وضع رمز، ينبغي تحديد ليس فقط مواصفات الرمز ، بل أيضا أين سنضعه. هذا المتطلب يؤدي إلى الترابط المباشر الموجود بين الرموز والتحويلات.

2.5.5 النمذجة بواسطة الرموز (Modeling With Symbols)

سيتم تعريف عدة رموز بصورة طبيعية جداً في فضاء WC، بينما الرموز الأخرى (كرمز المقاومة أو الشكل السداسي) عبارة عن أشكال قد لا تنتمي إلى منظومة إحداثيات معينة مرافقة معها. من أجل السماح لهذا التباين، سنقوم بإضافة منظومة إحداثيات جديدة تدعى إحداثيات رئيسية أو نمذجية (Master or Modeling Coordinates). لنأخذ بنظر الاعتبار رمز الحرف 'A'. نستطيع تعريفه وهو في مركز منظومة الإحداثيات الرئيسية مع أبعاد مناسبة. مثل هذه الرموز تكون أساس تطبيقات النشر بالحاسوب المنضدي [Post 85]. ينبغي تغيير أبعاد الرمز إلى الحجم المطلوب. وأيضا يجب توجيهه بصورة صحيحة بواسطة التدوير. وأخيراً، ينبغي وضعه عند الموقع المطلوب بواسطة الانتقال. هذه السلسلة

من العمليات، مبينة في الشكل 20.5، حيث نحصل على تحويل حالة (Instance Transformation)

I= TRS



الشكل 20.5 حالة مشاهدة

حيث أن S,R,T هي مصفوفات الانتقال، التدوير وتغيير الأبعاد. لاحظ ذلك، الترتيب الذي نقوم بتأدية هذه العمليات تكون حاسمة. إذا عرفنا الرموز لتكون متمركزة في فضاء الإحداثيات الرئيسية، قد يكون هذا الترتيب هو الأفضل لمعظم التطبيقات.

لنعود إلى مثالنا السابق حول المقاومة، يبدوا أن إحدى الطرق لخلق حالة مشاهدة لرمز المقاومة (برهة واحدة) يكون من خلال الإجراء التالي:

resistor (matrix)

حيث أن matrix هي تحويل حال مشاهدة. أو قد نضرب مثلاً لحالة مشاهدة من خلال الإجراء التالي:

resistor (x,y, theta, alpha, beta)

الذي يعطي معلميات الانتقال التدوير وتغيير الأبعاد بصورة مباشرة.

قد يتطلب الأمر إلى آلية عامة وقابلة للتكيف أكثر عندما يكون لدينا عدد كبير من الرموز أو نرغب في إضافة رموز جديدة. لنفترض تم تعريف كل رمز برقم. إذن، يمكننا إدراك جميع رموزنا من خلال استدعاء واحد للإجراء:

symbol (sym-num, instance – matrix)

الذي يقوم بتطبيق تحويل حالة مشاهدة (Instance Transformation) على الرمـــز المشار إليه. نستطيع كتابة الإجراء رمز symbol مستخدمين بعض الإجراءات الموجودة في حزمتنا لبرامج التحويل. بالنسبة لتطبيق تصميم دائرة كهربائية ، قد تبدوا الدالـة symbol مشابه إلى ما يلي:

```
void symbol (sym-num, instance – matrix)
int sym-num;
Matrix instance- matrix;
#define RESISTOR1
#define CAPACITOR 2
#define CHIP 3
        :
            switch (sym-rum)
            {
            case RESISTOR:
                for (i= 0; i < num- resist – pts; i++)
                    transform – point (...............
            break;    gpolyline (..............
        case CAPACTOR:
            :
```

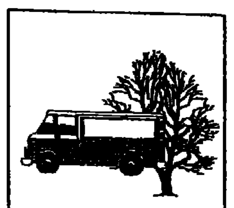
توجد هنالك عدد من الطرق البديلة لتنفيذ الرمز ووسائل إظهار حالـــة مشـــاهدة. سوف لا نمعن النظر في هذه البدائل. لكن بدلاً من ذلك سننتقل إلى دراسة مشكلة تكوين نماذج متعددة المستويات أو نماذج هرمية (Hierarchical Model).

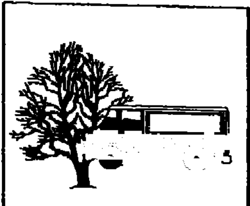
6.5 النمذجة بواسطة العلاقات (Modding With Relationships)

لحد الآن، قمنا باستخدام التحويلات للنمذجة بأسلوب بسيط، أي معنى احتســـاب التحويلات التي نرغب بتطبيقها على كيان أولي معين أو رمز. هذه النظـــرة إلى نمذجـــة تطبيق، ذا طبيعة خطية. البرنامج الذي يصف نموذج كهذا سيتكون من قائمـــة طويلـــة

لحسابات التحويل والرموز وتنشيط أزواج من الرموز والتحويل. تمتاز معظم التطبيقـــات المهمة (أو المعقدة منها) بالترابط الموجود بين كياناتها الأولية ورموزها.

على سبيل المثال، لو أردنا توليد سلسلة من الصور لسيارة تتحرك عبر مشهد كمــا في الشكل 21.5.





الشكل 21.5 سيارة متحركة

معدل دوران الدولاب يحدد سرعة حركة جسم السيارة. من وجهة نظر الرسومات، تكون أفضل طريقة لتمثيل دواليب السيارة كأربعة حالات مشاهدة لرمـــوز الـــدولاب. ويمكن تمثيل الهيكل (Chassis) بعدئذ بصورة مستقلة كرمز إضافي واحد أو أكثر.

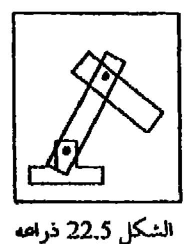
لو استخدمنا مثل هذا التمثيل، مع ذلك، فإن مبرمج التطبيق يكون هـــو المســـؤول لإدامة الترابط بين دوران الدولاب والحركة الأمامية للسيارة.

في هذا البند، سنقوم بدراسة الطريقة الممكنة في استخدام التحويلات للتعبير عن هذه العلاقات. سيكون مثالنا هو ذراع إنسان آلي بسيط (Simple Robot Arm). وفي البنـــد القادم، سنعطي لهذه التقنية شمولية أكثر من خلال نماذج ذات هيـــاكل شـــجرية (Tree-Structured Models).

1.6.5 ذراع إنسان آلي بسيط (A Simple Robot Arm)

لنأخذ ذراع آلي بسيط كما هو مصور في الشكل 22.5. يقبل الحركة في بعديـــن (Two Dimensions). يتكون ذراع الإنسان الآلي هذا من :

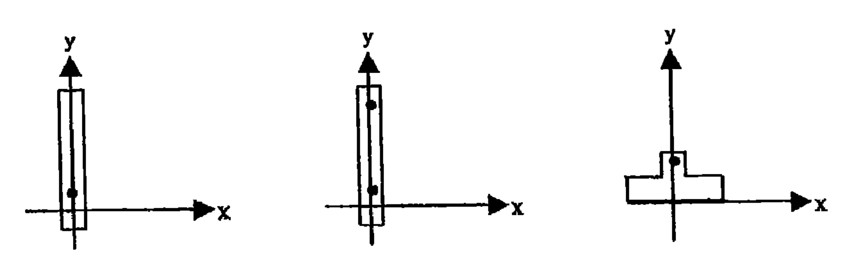
1. ذراع سفلي متصل بواسطة محور إلى قاعدته.
2. ذراع علوي مرتكز بواسطة محور على الذراع السفلي.



الشكل 22.5 ذراعه الإنسان الآلي

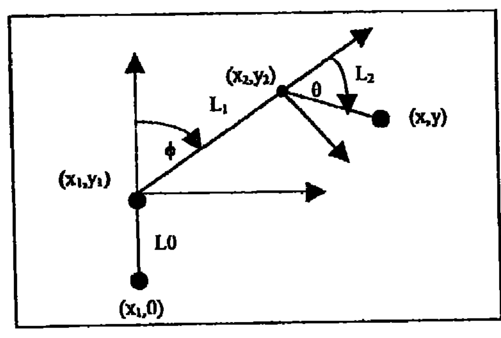
تتم السيطرة على آلية حركة الذراع بواسطة الدوران حول المحاور. هكذا، فعندما نقوم بتحديد زوايا التدوير، يكون موقع الذراع قد تم تعينه. باستطاعتنا استخدام هذا الذراع البسيط كنموذج جهاز تحديد موضع ثنائي الأبعاد وذلك بواسطة ربط إبرة أو مجس عند نهاية الذراع، أو نجعله يعمل في ثلاثة أبعاد وذلك بالسماح للقاعدة أن تدور حول المحور-z.

إننا نرغب في نمذجة الذراع لكي يتيح لنا القيام بالمناورة مع الإنسان الآلي بسهولة، أو لربما نقوم بتغيير تصميمه. من الممكن تسهيل الوصول إلى هذا الهدف وذلك بتعريف ثلاثة رموز لكل من القاعدة، الذراع السفلي والذراع العلوي، كما هو مبين في الشكل 23.5.



الشكل 23.5 استخراج معادلات الإنسان الآلي

لقد تم اختيار منظومة الاحداثيات الرئيسية بحيث كل رمز يتم مركزته على المحور العمودي ويستقر على المحور الأفقي. سنقوم برسم الإنسان الآلي بواسطة استخدام تحويلات حالة المشاهدة من أجل وضع الأجزاء في مواقعها المناسبة. لاحظ ذلك، أن كل جزء في نموذجنا يمكن رسمه بواسطة متعدد خطوط. أن مشكلتنا ستكون كيفية وضع الأجزاء في مواقعها الصحيحة نسبة الواحد للآخر، طريقة تغيير هذه المواقع أثناء حركة الذراع.



الشكل 24.5 استخراج معادلات الإنسان الآلي

نستطيع استخراج معادلات الإنسان الآلي في عدة طرق. كطريقة مباشرة نقوم برسم الإنسان الآلي كما مبين في الشكل 24.5، حيث يمكننا وصف الإحداثيات كما يلي:

1. استقرار القاعدة على الأرض مع مركزه العمودي عند النقطة $(x_1,0)$.
2. يكون طرف الذراع العلوي عند النقطة (x,y)
3. يكون موقع محور ارتكاز القاعدة عند النقطة (x_1,y_1)
4. يكون محور الارتكاز بين الذراعين العلوي والسفلي عند النقطة (x_2,y_2).

لنفترض أن L_1 تمثل المسافة بين نقطتي محور الارتكاز على الذراع السفلي و L_2 تشير إلى المسافة بين نهاية الذراع العلوي ونقطة محور ارتكازه. جميع هذه القيم ممكن تمهيلها ببساطة من القياسات المبينة في الشكل 24.5 وقيم الزوايا θو ϕ .

نستطيع تصور شكل العصا (Stick Figure) هذا كمخطط هيكلي للإنسان الآلي، وأن وصف حركته تتيح لنا بسهولة نمذجة حركة الهيكل بصورة كاملة. إن المجموعتين من المعادلات التالية التي تصف هذا الهيكل يمكن كتابتهما بواسطة استخدام مثلثات بسيطة كما يلي:

$$x_2 = x_1 - L_1 \sin(\phi),$$
$$y_2 = y_1 + L_1 \cos(\phi),$$
$$x = x_2 - L_2 \sin(\theta+\phi),$$
$$y = y_2 + L_2 \cos(\theta+\phi),$$

من الواضح موضع الطرف (x,y) هو دالة لكلتا الزاويتين، وهكذا يعكس الاقتران أو الترابط الموجود بين العناصر. مع أنه صحيح، لو قلنا أن هذه المعادلات لا تظهر العلاقة بصورة جيدة لنموذجنا الذي قام بتجزئة الإنسان الآلي إلى ثلاثة أجزاء. أيضاً هذه المعادلات لا تساعدنا في توليد سلسلة من صور الإنسان الآلي أثناء تغيير هاتين الزاويتين.

2.6.5 النمذجة بواسطة مصفوفات التحويل

(Modeling with Transformation Matrices)

نستطيع معالجة المشكلة من وجهة نظر مصفوفات التحويل. دعنا نعيد النظر في دراسة معادلاتنا مع فكرة تمثيل جميع النقاط على الإنسان الآلي، لكي يتسنى لنا من رسم الإنسان الآلي بواسطة منظومتنا للرسومات. يتم تحريك كلا الذراعين الأعلى والأسفل بواسطة التدوير. لنفترض اعتبار نقطة ارتكاز المحور السفلي لكل ذراع هي نقطة أصل

منظومة إحداثيات رئيسية خاصة بها. إذن تكون حركة كل جزء من الذراع هي دوران بسيط حول نقطة أصله.

من أجل الحصول على الموضع الحقيقي للذراع الأسفل في WC ، ينبغي علينا نقل نقطة الأصل إلى النقطة (x_1,y_1). إذا كان طول القاعدة Lo اعتباراً من الأسفل إلى نقطة ارتكاز المحور، فإن مصفوفة الانتقال هذه هي:

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & Lo \\ 0 & 0 & 1 \end{bmatrix} = T_1\,(0,Lo)$$

يتم تدوير الذراع الأسفل بواسطة مصفوفة تدوير قياسية $R_1(\phi)$ ، لذا تكون مصفوفة حالة المشاهدة للذراع الأسفل هي :

$$A = T_1\ R_1$$

قبل أن نقوم باستخراج علاقة مشابهة للذراع الأعلى، لنرى ماذا يحدث إذا حركنا القاعدة. باستطاعتنا تعريف مصفوفة انتقال T التي يمكن بعدئذ تطبيقها على كل نقطة تعرف القاعدة. مع أن تحريك القاعدة أيضا تؤدي إلى حركة الذراعين الأسفل والأعلى. إذن، النقاط على هذين الهيكلين ينبغي أيضاً نقلها بمقدار T. الآن بالنسبة للذراع الأسفل، يكون تحويل حالة المشاهدة معطاة بواسطة TA. إذن، تفسير مفيد للمصفوفة A هي أنها تقوم بمركزه الذراع الأسفل نسبة إلى القاعدة وهكذا يتم توحيد العلاقة بين هذين الشيئين المنظورين.

لو واصلنا هذا النقاش، يمكننا وصف الذراع العلوي نسبة إلى الذراع السفلي وباستخدام تحديد الموقع النسبي للذراع السفلي نسبة إلى القاعدة ليساعدنا في الحصول على الموقع الحقيقي للذراع العلوي. يتم تدوير الذراع العلوي حول نقطة أصله بواسطة مصفوفة التدوير $R_2\,(\theta)$. ونسبة إلى الذراع السفلي، يتم نقله أولاً بواسطة المصفوفة $T_2\,(0,L_1)$. وبضم هاتين المصفوفتين، نحصل على المصفوفة:

$$B = T_2\, B_2$$

الآن بوضع جميع هذه الحسابات معاً، نقوم بتعيين الموقع المطلق للـــــذراع العلـــوي بواسطة تسلسل (Concatenating) هذه المصفوفات لتكوين المصفوفة TAB.

3.6.5 تحريك النموذج (Animating the Model)

افترض أننا نرغب تحريك الإنسان الآلي وذلك بتغيير مواقـــع الذراعـــين العلـــوي والسفلي. اذن تحتاج تغيير مصفوفتين للتدوير فقط وإعادة احتساب المواقع. وما يلي هـــو جزء من البرنامج، مفترضين أن لدينا إجراء الرمز symbol (matrix,n) الذي يرسم الرمز n مستخدماً المصفوفة matrix لتحويل حالة المشاهدة. تكون رموزنـــا UPPER-ARM. BASE, LOWER ARM.

```
/* Define Initiol Rotation and Transaltion Matrices */
static Matrix rl=
     { 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0);
static Matrix r2=
     { 1.0 , 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0};
static Matrix to =
     { 1.0, 0.0, x0, 0.0, 1.0, y0, 0.0, 0.0, 1.0};
static Matrix t1=
     { 1.0, 0.0, 0.0, 0.0, 1.0, L.0, 0.0, 0.0, 1.0};
static Matrix t2=
     { 1.0, 0.0, 0.0, 0.0, 1.0, L1, 0.0, 0.0, 1.0};
Matrix a , b , m;
int i;
/*Draw Robot Procedure*/
draw-robot ( )
{
     symbol (BASE, t0);
     ac- matrix (t1, r1, a);
     ac-matrix (t0, a ,m);
     symbol (LOWER, ARM,m);
     ac-matrix (t2, r2, b);
```

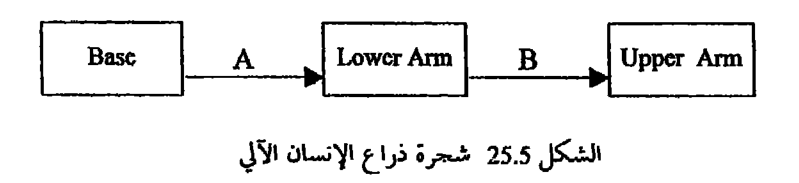
```
        ac-matrix (m, b, m);
        symbol (UPPER – ARM, m);
    }
    /* Draw Robot in Its Initial Position*/
    draw – robot (  );
    /Increment Rotation Matrices and Redraw*/
    for (i = o; i < NTIMES; i++)
    {
        ac – rotate (THETA, r1);
        ac – rotate (PHI, r2);
        draw-robot (  );
    }
```

7.5 استخدام التدرج (هيكل هرمي) والتكرار المتداخل (المعاودة):
(Using Hierarchy and Recursion)

يبين نموذج الإنسان الآلي اعتمادية الجزء الواحد من النموذج على غيره من الأجزاء. حتى هذه المرحلة تركنا إلى برنامج المستفيد للتعبير عن هذه الاعتماديات وإرسال تحويلات حالة المشاهدة إلى إجراء الرمز (symbol routine). في هذا البند سنقوم بدراسة أولية لاستخدام الأشجار (Trees) لتمثيل العلاقات. إن لغات البرمجة كلغة C وباسكال تتيح للمستفيد بتعريف ومعالجة الشجرات من خلال استخدام تراكيب (أو قيود) وبرمجة التكرار المتداخل أو المعاودة (Recursive Programming).

1.7.5 ذراع الإنسان الآلي كشجرة (The Robot Arm as a Tree)

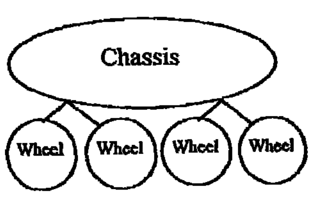
إن العلاقات الموجودة ما بين أجزاء ذراع الإنسان الآلي يمكن تمثيلها كما مبين في الشكل 25.5.



الشكل 25.5 شجرة ذراع الإنسان الآلي

هذا الهيكل هو عبارة عن نوع خاص من مخطط بياني (Graph) يعـــرف بالشـــجرة (Tree)، تتكون الشجرة من نقاط تدعى العقد (Nodes) أو الرؤوس (Vertices) متصلـــة بقطع خطوط تدعى الحافات (Edges). كل عقدة في الشجرة، باستثناء العقدة العليـــا أو الجذر (Root Node)، لها عقدة أب (Parent Node) واحدة فقط. كل عقدة تكون لهـــا عقد الأبناء (Child Nodes). العقد التي لا تحتوي على أبناء يطلق عليها عقـــدة طرفيـــة (Terminal Node). ضمن هذا التركيب، تكون لدينا مرونة هائلة في كيفية اســـتخدام العقد والحافات.

نستطيع البدء بتخصيص مصفوفة تحويل لكل حافة ورمز لكل عقدة كما مبـــين في الشكل 25.5. الرموز عند العقدة هي تلك الرموز التي نود رسمها. المصفوفـــات كمـــا تم تعريفها في البند السابق هي تلك التي تقيم علاقة بين الموقع لكيان أولي عند عقدة واحـــدة مع تلك الكيانات الأولية عند عقدة أبيه. جميع المعلومـــات حـــول ذراع الإنســـان الآلي موجودة في الشجرة. نستطيع تعيين موضع القاعدة بواسطة تهيئة مصفوفـــة T، الـــتي تم استخدامها في تحديد جميع تحويلاتنا لحالة المشاهدة في البند السابق. ومن أجل رســـم ذراع الإنسان الآلي، علينا اجتياز (Traverse) الشجرة، هذا يعني، يجب تتبع الشجرة ابتداءً مـــن عقدة الجذر مروراً بجميع الأبناء. أثناء اجتيازنا للتركيب (Structure) نقـــوم بسلســـلة (Concatenated) المصفوفات للحصول علـــى تحويـــل حالـــة مشـــاهدة (Instance Transformation) صحيحة لكل رمز نجده عند العقدة. هكذا، عند عقدة الجذر، لدينا T فقط، التي تمثل تحويل حالة المشاهدة للقاعدة. وعند العقدة الثانية، نحصل على TA الـــذي يتم تطبيقه على الذراع الأسفل وأخيراً، عند العقدة الثالثة، نحصل على TAB.



الشكل 26.5 السيارة

الآن ندرس كيف يمكننا إجراء هذه العمليات مـــن داخل برنامج المستفيد. توجد هنالك قضيتين منفصلتـــين ولكن بينهما علاقة:

1. كيف نقوم بتمثيل شيء منظور في نموذج الشجرة.
2. كيف نقوم باستعراض أو اجتيـــاز (Traverse) هذه الشجرة؟

نحن نرغب في استخدام أشجار قد تكون أكثر تعقيداً مما هو عليه في مثالنا لـــذراع الإنسان الآلي البسيط. على سبيل المثال لنأخذ بنظر الاعتبار، سيارة متحركة . قد يكـــون هيكل شجري محتمل كما مبين في الشـــكل 26.5.

Body
Arm | Arm | Leg | Leg

حيث افترضنا فيه كل إطار له شـــجيرة (Subtree) للهيكل (الشاصي– Chassis). بالنسبة لنمـــوذج بسيط لجسم الإنسان، قد نستعمل شـــكل العصـــا المبين في الشكل 27.5.

الشكل 27.5 شكل العصا

من الممكن بسهولة جعل هذا الهيكل البســـيط أكثر تطوراً. النقطة الأساسية هي تلك الحاجـــة إلى تمثيل قابل للتكيف الذي سيتيح اســـتخدام عـــدد اختباري من الأبناء لعقدة معينة. لو نلاحظ أن الشجرة تحتوي على عدد من الشـــجيرات، وكل واحدة متصلة بأبيها، لهذا نستطيع أن تدرك إمكانية اســـتخدام تعـــاريف التكـــرار المتداخل (Recursive Definition) وخوارزماتها.

2.7.5 تمثيل شجرة (Representing a Tree):

Symbol
Matrix
Node-list

Child-1
Child-2
.
.
child-n

الشكل 28.5 تركيب الشجرة

سوف نستخدم تركيـــب (Structure) كـــالمبين في الشكل 28.5 لتمثيل شجراتنا.

كل عقدة على الشجرة ستحتوي علـــى مؤشـــر إلى قائمة فيها جميع الأبنـــاء لتلـــك العقـــدة، بالإضافـــة إلى المعلومات الضرورية لرسم الرمز المرتبط مـــع العقـــدة .

وباستخدام التراكيب من خلال لغة C، يمكننا تعريف عقدة شجرة كما يلي:

```
Struct tnode
{
    int symbol;
    Matrix matrix;
    struct list *nodelist;
};
```

هنا كل عقدة تحتوي على :

1- رقم الرمز الذي سوف يرسم عند تلك العقدة.

2- مؤشر إلى مصفوفة تحديد الموضع الذي يقوم بوضع هذه العقـــدة بالنســـبة إلى أبيها. وهذه المعلومة ستكون في متناول جميع أبناء هذه العقدة.

3- مؤشر إلى قائمة موصولة (Linked List) لمجموعة عقد تمثل أبناء العقـــدة. قـــد تكون هذه القائمة لعقد الأبناء فارغة ولهذا قد يحتوي المؤشر على معلومة Null.

نستطيع تعريف القائمة الموصولة كما يلي:

```
struct list
{
      struct tnode *nodc;
      struct list *next;
};
```

كل عنصر في تركيب القائمة يتكون من مؤشرين هما:

مؤشر لعقدة ما في الشجرة.

مؤشر إلى العنصر التالي في القائمة.

هنا نحتاج إلى دالتين هما: tree ( ) و addlist ( ). هاتين الدالتين ضروريتـــين في تخصيص مساحة لخزن الشجرة بصورة ديناميكية (Dynamically) أي بصورة فعالـــة في حالة تغيير مستمر للشجرة:

```
struct tnode *tree ( )
{
      char *calloc ( );
      struct tnode  *p;
      p = (struct tnode *) calloc (1, sizeof (struct tnode));
      p → nodelist = NULL;
      return (p);
}
struct list *addlist ( )
```

```
}
    char *calloc ( );
    struct list *p;
    p = (struct list *) calloc (1, sizeof (struct list ));
    p → next = NULL;
    p → node = NULL;
    return (p);
}
```

هنا، لقد استخدمنا برامج خدمية (Utilities) قياسية في لغة C وهي sizeof لإعادة مقدار من مساحة الخزن المطلوبة وبرنامج خدمي آخر هو calloc لتخصيص هيكل واحد من هذا النوع. يقوم هذين الإجراءين بالحصول على عنوان المساحة الضرورية للذاكرة الحرة وإضافتها إلى الشجرة أو إلى القائمة وأيضاً وضع المؤشرات NULLS في التركيب لكي يتمكن البرنامج من تحديد مواقع العناصر الطرفية (Terminal Elements) أينما وجدت.

باستخدام هذه التراكيب، يمكننا تعريف شكل العصا بأسلوب مباشر (وإن يكن قبيحاً)

```
root = tree ( );
root → symbol = BODY;
root → matrix = body;
root → nodelist = addlist ( );
root → nodelist → next = addlist ( );
root → nodelist → next → next = addlist ( );
root → nodelist → next → next→ next = addlist ( );
root →  nodelist → node = tree (  );
root → nodelist → next → node = tree ( );
root → nodelist → next → next→ node = tree ( );
root→ nodelist→ next→ next→next→ node= tree( );
root → nodelist → node → symbol = ARM;
root → nodelist → next → node → symbol = ARM;
root → nodelist → next → next →  node→ symbol= LEG;
```

```
root → nodelist → next → next → next→ node → symbol = LEG;
root → nodelist → node → matrix = arm1;
root → nodelist → next → node → matrix = arm2;
root → nodelist → next → next → node → matrix = leg1;
root → nodelist → next → next → next → node → matrix= leg2;
```

هنا تكون BODY ، ARM، LEG هي أسماء الرموز التي نفترض تم تعريفـــها في إحداثيات رئيسية. وأما المصفوفات arm2, arm1, body و leg2 , leg1 تقوم بوضــــع هذه الرموز نسبة إلى عقد أبائها. إن قطعة البرنامج هذه تبين بصورة صريحة المســـارات في الشجرة التي نستخدمها للوصول إلى كل رمز (ولكنه غير جميل أو بارع) . سنقوم بإعطاء نسخة أفضل من قطعة البرنامج هذه في البند القادم.

3.7.5 اجتياز النموذج (Traversing the Model)

مع أن وصفنا لشكل العصا كان غير أنيقاً، لكن الإجراء لاجتياز الشجرة مســتخدماً تكرار متداخل (Recursion) يكون بسيطاً جداً. سنقوم باســتخدام مفــهوم مصفوفــة التحويل الحالي (CTM- Current Transformation Matrix) . إنها المصفوفة التي ســوف نستخدمها عند عقدة معينة لرسم الرمز (إذا كان هناك أي رمز) الذي نجده هناك.

نقوم برسم الشجرة بواسطة إجراء رسم الشجرة draw-tree، الذي تنقـــل إليــه مصفوفة الموقع pos-matrix لتحديد موضع الشجرة بأكملها مع مؤشر إلى عقدة الجـــذر للشجرة. يقوم هذا الإجراء برسم أي رمز يجده عند عقدة الجذر وذلك بسلسلة المصفوفـــة المرره إليه مع المصفوفة الموجودة عند تلك العقدة أولاً. ويتم وضع الناتج في المصفوفـــة CTM مستخدماً إجراء تراكم مصفوفة ac-matrix. لقد قمنا بالاحتفاظ برقم الرمـــز 'o' ليعني أنه لا يوجد هنالك رمز للرسم عند هذه العقدة. نقوم برسم الرموز بواسطة تمريــــر رقم الرمز والمصفوفة CTM إلى إجراء رسم الرموز draw-symbol. إذا لم تكـــن قـــائم عقدة الأبناء فارغة، علينا الأخذ بنظر الاعتبار جميع عقد الأبناء وهذا يتم فعلاً من خـــلال دالة اجتياز القائمة (traverse list). تستخدم هذه الدالة التكرار المتداخل (Recursively) للإجراء draw-tree لرسم جميع الشجيرات الفرعية (Subtrees) الموجـــودة في قائمتـــه. يكون البرنامج كما يلي:

```
draw-tree (pos-matrix, root)
struct tnode * root;
Matrix pos-matrix;
{
    Matrix ctm;
    ac-matrix (pos-matrix, root → matrix, ctm);
    if (root → symbol ! = o) draw – symbol ( ctm, root → symbol);
    if (root → nodelist ! = NULL)
    traverselist (ctm, root → nodelist);
}
traverselist (pos- matrix, list –root)
Matrix pos-matrix;
struct list *list-root;
{
    draw – tree (pos – matrix, list – root → node);
    if (list – root → next ! = NULL)
        traverselist (pos-matrix, list – root → next);
}
```

إن الطريقة المعينة لعملية الاجتياز التي تم استخدامها (هنالك عدة طرق) تبدأ بتحديث مصفوفة التحويل الحالية (CTM) التي تم تمريرها وذلك بواسطة عملية تراكمها مع المصفوفة عند تلك العقدة. إذا وجد هناك رمز عند العقدة نقوم برسمة. وإذا كان لدى العقدة أية أبناء ، فالقائمة الموصولة لعقد الأبناء عندها يتم تطبيق نفس الإجراء على كل عقدة من عقد الشجرة لهذه القائمة.

يمكننا تتبع عملية رسم الإنسان الآلي وذلك بتتبع مسار استدعاء الإجراء مثل:

```
draw – tree (ctm, robot);
```

حيث ctm هو مؤشر إلى مصفوفة التي تحدد موقع التركيب بأكمله في العالم. لنسمي هذه المصفوفة بـ M، وفي البداية قد تكون M مصفوفة محايدة (identity matrix). أولاً يقوم إجراء الاجتياز تبديل M ووضع MTبدلها وتستخدم المصفوفة الأخيرة MT لرسم

القاعدة. يوجد لهذه العقدة ابن، وحيث يتم تمرير المصفوفة MT. نقوم باجتياز شجيرة الابن وذلك بتحديث المصفوفة CTM إلى MTA أولاً، ويتم رسم الذراع السفلي. هذه العقدة لها ابن أيضاً، وهكذا يتم تمرير المصفوفة CTM، التي تحدث إلى MTAB قبل رسم الذراع العلوي.

4.7.5 مناقشة (Discussion)

يقوم إجراء الاجتياز برسم الشجرة من العقد العليا إلى السفلى. إذا احتوت جميع عناصر رسومنا البيانية في النهاية على متعددات خطوط أو أنواع أخرى غير ممتلئة، ستكون الصورة الناتجة هي تقريباً نفسها، بصرف النظر عن خوارزمية الاجتياز. أما إذا استخدمنا مساحات ممتلئة ، يكون الترتيب الذي تم فيه رسم الكيانات الأولية أمر حاسم، ولربما نتوقع خوارزميات اجتياز مختلفة لإنتاج صور مختلفة. وكذلك نلاحظ أن التكرار المتداخل لاجتياز الشجرة (Recursive Tree Traversal) قد يكون بطيء، لكن هنا يكون تركيزنا على النمذجة بدلاً من كفاءة البرنامج النهائي. من الناحية العملية، يمكننا دائماً تبديل خوارزمية اجتياز الشجرة ذات التكرار المتداخل بخوارزمية تكرارية (Iterative).

بالرغم أن هذه الطريقة تكون نوعاً ما عامة، لكن توجد هنالك عدة طرق لتمثيل تراكيب هرمية. في كثير من التطبيقات، قد نحتاج إلى تراكيب بيانات أكثر تفصيلاً. سوف نقوم بدراسة تركيب آخر – مخطط غير حلقي Acyclic Graph – لاحقاً في هذا الفصل. لقد أثبتت النماذج الهرمية لتراكيب الشجرة (Tree-Structured Hierarchical Models) أسلوباً لا يثمن في النمذجة. بالرغم من أننا استخدمنا التكرار المتداخل وتراكيب بيانات بسيطة لكنها ليست من الضروري أن تتواجد في اللغة. بالضبط كما قمنا ببناء مجموعة روتينات لمصفوفات تحويل، كان بمقدورنا بناء روتينات لتمثيل ومعالجة الشجرات دون اللجوء إلى استخدام إمكانيات لغةC. مع ذلك، هذه الإمكانات فعلاً تجعل البرمجة أكثر سهولة وبالتأكيد تؤدي إلى تطبيقات أكثر كفاءة.

وأخيراً، نلاحظ أننا نظرنا إلى النمذجة كعمل يقوم به برنامج التطبيق. وحيث أنها فرضت على المستفيد بسبب النقص الموجود في بعض الإمكانات عالية المستوى في

منظومـــة GKS. إن منظومـــة هرميـــات المـــبرمج القياســـي للرســـومات المتفاعلـــة (Programmer's Hierarchical Iterative Graphics Standard – PHIGS) هي عبـارة عن قياسيات تجسد عديد من هذه الأفكار التي عرضناها تواً. بصـــورة خاصـــة، يكـــون مفهوم مصفوفة التحويل الحالي هو جزء من المنظومة والقطع (تدعى تراكيب في منظومـــة PHIGS)، تتيح لها الإشارة إلى قطع أخرى (تراكيب أخرى) التي تقوم بـــتزيد المســـتفيد بطريقة بسيطة لتركيب نماذج هرمية.

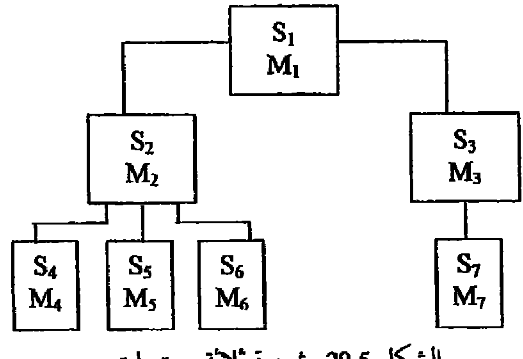
8.5 تنفيذ أنواع بيانات تجريدية

(Implementation Of Abstract Data Types)

إن دراستنا السابقة حول النماذج الهرمية تؤكد على استخدام الأشجار في نمذجـــة تطبيقات الرسومات. في هذا البند سنؤكد على الفرق بين النموذج المفاهيمي وتنفيذ ذلـك النموذج في منظومة أو لغة برمجة معينة. كنتيجة لهذه الدراسة ، ســـنعيد تصميـــم مثالنـــا السابق بأكثر أناقة.

1.8.5 العمليات على الشجرة (Operations On a Tree)

نبدأ مع مثال الشجرة في الشكل 29.5، الذي يتكون من ثلاث مستويات.



الشكل 29.5 شجرة ثلاثة مستويات

نفترض أن كل عقدة تحتوي على رقم الرمز ومصفوفة تحدد لنا طريقة وضـــع هـــذا الرمز نسبة إلى موقع عقدة الأب. بالأساس ، هذه الطريقة هي نفس تلك التي استخدمناها في البنود السابقة لتعريف أنواع من تراكيب الشجرات التي تكون نافعـــة في تطبيقـــات

الرسومات. من الممكن تنفيذ هذه الشجرة كما في السابق باستخدام قائمة موصولة للأبناء عند كل عقدة. بدلاً من ذلك، سنتخذ أسلوب مختلف.

عند هذه المرحلة لدينا نوع من البيانات وهي الشجرة . نحن نرغب إجراء عمليات على هذا النوع من البيانات. عند هذا المستوى من التجريد (Abstraction)، نحن نرغب وصف هذه العمليات بلغة نوع البيانات التجريدية (Abstract Data Type-ADT) بدلاً من لغة التنفيذ. لو تمكنا من وصف العمليات بصورة مستقلة عن أي تنفيذ، لاستطعنا كتابة برامج عالية المستوى وتترك قضايا التنفيذ إلى خصوصيات تحقيقها أو إلى خصوصيات الأجهزة المنفذة. إلى حد ما هذا الفصل بين العمليات وطريقة التنفيذ هو بالضبط ما يحصل في حزمة برامج الرسومات. نحن نعلم بطبيعة الحال ماذا تكون وظيفة وعمل الدوال وما تحتاجه من مدخلات وما ستنتجه من مخرجات. لذلك مبرمجي التطبيقات لا يحتاجون معرفة العمل الداخلي لهذه الإجراءات. بالنسبة لتطبيقات النمذجة، نحن نرغب إجراء نفس التقسيم. لكن لسوء الحظ، نظراً لكون النمذجة الهرمية هي ليست جزء من منظومة GKS، لهذا فرض علينا أن نظهر قضايا التنفيذ. مع هذا، نحن نؤكد على الفصل بين نوع البيانات وطريقة تنفيذها.

لنعود إلى مثالنا، نلاحظ هنالك وجود جزئين رئيسين لطريقة نمذجتنا مع الأشجار هما:

1- وصف الشجرة 2- رسم الشجرة

فيما يلي تعريف لكل جزء.

أولاً: تتضمن عملية وصف الشجرة إضافة عقد إلى الشجرة لتكوين شجرة أكثر تعقيداً. نستطيع وصف هذه العملية من خلال الإجراء (عقدة جديدة – (new-node):

new – node (root, matrix, symbol)

سيقوم هذا الإجراء بإعادة مؤشر إلى عقدة تعتبر ابن العقدة root، مع المصفوفة matrix والرمز symbol تم خزنهما في العقدة الجديدة. قد تكون عبارات البرنامج الذي يصف مثالنا الشجرة مشابه بعض الشيء إلى ما يلي:

```
root= new- node (NULL, 1, m1);
child 1 = new = node (root, 2, m2);
```

```
child 2 = new-node (root, 3 , m3);
grandchild1 = new-node (child1,4,m4);
grandchild2 = new – node (child1, 5, m5);
grandchild3 = new- node (child1, 6,m6);
grandchild4 = new-node (child2, 7, m7);
```

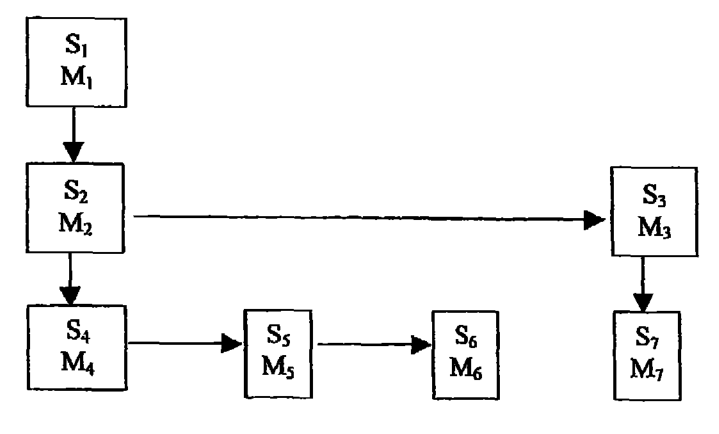
ثانياً: لرسم الشجرة تحتاج استدعاء الإجراء مرة واحدة كالآتي:

draw-tree (position, root)

الذي هو بالأساس الإجراء الذي استخدمناه من قبل . هنا يعتبر root مؤشر إلى شجرة و position هي المصفوفة المستخدمة لوضع الشجرة بأكملها في الموقع المطلوب.

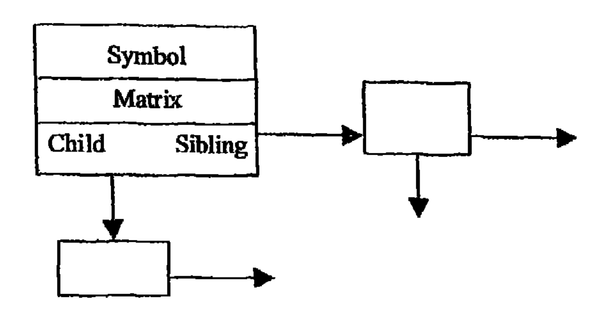
2.8.5 تنفيذ آخر (Another Implementation)

ينبغي أن يكون من السهولة تنفيذ هاتين الدالتين مستخدمين النموذج الخاص في البند السابق-تركيب عقدة وقائمة موصولة. من أجل التأكيد على الخاصية التي تكون فيها التنفيذ مستقل عن نوع البيانات وعملياتها، سنقوم بدراسة تنفيذ يختلف بعض الشيء. سوف نستخدم ما يسمى غالباً شجرة الابن لليسار -والأخ لليمين (left child – right sibiling tree). بالإمكان تنفيذ هذه الشجرة بواسطة تركيب بيانات منفردة الذي يحتوي نفس العدد من العناصر عند كل عقده. نستطيع فهم التركيب وذلك الأخذ بنظر الاعتبار الشكل 30.5.



الشكل 30.5 شجرة الابن لليسار – والأخ لليمين

لو قمنا بعَد المؤشرات الفارغة (Null Pointers)، لوجدنا أن كل عقــدة في هــذه الشجرة تحتوي بالضبط على أخ واحد وابن واحد. بعد ملاحظة دقيقة يظــهر أن هــذه الشجرة تحتوي بالضبط على نفس المعلومات التي تحتويها الشجرة في الشكل 28.5. وهــذا يقترح التنفيذ المصور في الشكل 31.5.



في لغة C يمكن وصف التركيب المطلوب كما يلي:

```
struct  thnode
{
      int symbol;
      Matrix matrix;
      struct tnode *child;
      struck tnode *sibiling;
}
```

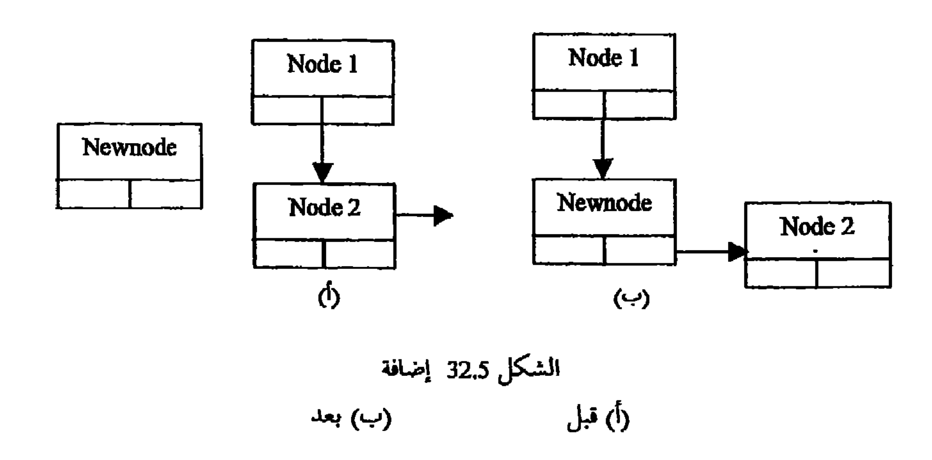
الآن نستطيع استخدام هذا التركيب لتنفيذ الإجرائين new- node و draw-tree.

فيما يلي وصف لهذين الإجرائين:

1- يقوم الإجراء new –node بإعادة مؤشر إلى العقدة الجديدة مع الرمز ومصفوفتـه المطلوبة عند هذه العقدة، الجزء الأول الضروري من العملية يكون مشابه إلى ما فعلنا مــن قبل. حيث نحصل على المساحة الضرورية من الذاكرة من خلال استدعاء دالة تخصيــص ذاكرة كالدالة calloc. بعدئذ نقوم بوضع الرمز ومصفوفته المطلوبة عند هــذه العقــدة وجعل المؤشرين اليسار واليمين NULL. في هذه المرحلة، لم يتم تنسيب هــذه العقــدة الجديدة إلى عقدة الأب. توجد هنالك حالتين هما:

أ- إذا كانت عقدة الاب لا تحتوي على أبناء، يكون مؤشر ابنه NULL. في هـذه الحالة، نستطيع ببساطة تبديل المؤشر NULL بالمؤشر إلى العقدة الجديدة.

ب- أما إذا كان مؤشر الابن ليس NULL ، معنى هذا أن عقدة الأب تحتوي على ابن سابق، لذا يجب إضافة أو إدخال (Insert) العقدة الجديدة. يمكننا تحقيق عملية الإضافة كما مبين في الشكل 32.5.



الشكل 32.5 إضافة

(أ) قبل (ب) بعد

وذلك بنقل المؤشر من الأب ليشير إلى العقدة الجديدة وجعل مؤشر الأخ في العقد الجديدة يشير إلى أخيه. هناك يوجد تعقيد واحد وأخير . إذا كان مؤشر الشجرة المار إلى الإجراء هو NULL، فإن العقدة التي تم تحديثها ستحل محل العقدة NULL. ما يلي هي قطعة البرنامج:

```
struct tnode *new – node (t,s,m)
struct tnode *t;
int s;
Matrix m;
{
    struct tnode *p;
    char *colloc ( );
    p= (struct tnode * ) calloc (1 , sizeof (struct tnode));
    p → symbol = s;
```

```
        p → matrix = m;
        p → child = NULL;
        if ( t= = Null)
    {
           p → sibling = NULL;
               t → child = p ;
               return (p);
        }
    }
```

2- سيكون إجراء رسم الشجرة draw-tree مشابه إلى الطريقة التي سبق استخدامها في البند السابق، ولكن مع تغيير ثانوي – حيث أن جميع الأخوة يستخدمون نفس المصفوفة التي تم نقلها من الأب. تكون طريقتنا الأساسية للاجتياز هي رسم الرمز عند عقدة، باستخدام الإجراء رسم الرمز draw-symbol. وبعدئذ اجتياز كلتا الشجرتين للإخوه والأبناء. يكون البرنامج هنا كما يلي:

```
draw – tree (m,p)
struct tnode *p;
Matrix m;
{
        float ctm [2] [3];
        ac-matrix (m, p → matrix, ctm);
        draw – symbol (ctm, p → symbol);
        if (p → right ! = NULL) draw – tree (m, p → sibling );
        if ( p → left ! = NULL) draw – tree (ctm, p → child);
}
```

لو استخدمنا المثال شكل العصا (Stick Figure) الذي تم التعرف عليه في البند السابق، هذا الإجراء سيقوم برسم الشجرة. إننا سنفترض ، لغرض التوافق، أن الإجراء draw- symbol يقوم بتفسير رقم الرمز '0' ليعني عدم وجود رمز للرسم عند العقدة. سيكون وصف العصا مستخدمين نفس المصفوفات والرموز، كما يلي:

```
root = new - node (NULL, BODY, bod);
child1= new-node (root, ARM, arm1);
Chid2= new – node (root, ARM, arm2);
```

```
child3 = new- node (root, LEG, leg2);
child4 = new - node (root, LEG, leg2);
```

مرة ثانية، نلاحظ أن هناك توجد طرق عديدة أخرى لتنفيذ تراكيب هرمية. في تطبيق معين، علينا دائماً موازنة الخواص المميزة التي توفرها منظومة الرسومات مقابل الخواص المميزة الموجودة في لغة البرمجة. في مثالنا، قمنا بالتخلص من بعض تحديدات منظومة GKS وذلك باستخدام بعض الخواص المميزة للغة C. وأما من الناحية الثانية، لو كنا نستخدم منظومة رسومات كمنظومة PHIGS التي توفر لنا تراكيب هرمية، لكننا نستخدم الترابط مع لغة فورتران بدون أية صعوبة. لو كان لدينا فقط ترابط مع لغة فورتران في منظومة GKS وكانت رغبتنا في تنفيذ تراكيب الشجرة، فإن إجراءاتنا : إضافة عقدة add-node ورسم شجرة draw-tree سوف تكون مختلفة.

9.5 من قطع إلى تراكيب (From Segment To Structures)

لحد الآن، قمنا بالتركيز على نماذج تراكيب الشجرة – (Tree Structured Models) . كل عقدة سواء كانت من نوع تجريدي (Abstract Type) أو نوع معرف لتنفيذ معين، سيكون لها تركيب ثابت– على سبيل المثال، مصفوفة ، رقم رمز ومؤشر واحد أو أكثر إلى عقد أخرى. يكون هنالك إعادة تفسير بسيط لما نقصده بالرمز الذي سيقودنا إلى أسلوب نمذجة قد يكون أكثر فعالية. أولاً سوف ننظر إلى كيفية الانتقال من نموذج في برنامج مستفيدنا إلى قطعة في منظومة رسوماتنا.

1.9.5 محتويات القطعة (Segment Contents)

في نظرتنا الحاضرة ، تكون النمذجة جزء من برنامج التطبيق. إذا استخدم برنامجنا منظومة كمنظومة GKS، سنقوم بالتالي تكوين قطع تحتوي على كيانات أولية معرفة من قبل نموذجنا. وبدون الخوض في دراسة قضايا التنفيذ التي سيتم مناقشتها في الفصلين القادمين، يمكننا إعطاء ملاحظتين هما:

1- المنظومة التي تفصل النمذجة من منظومة الرسومات بدون شك تكون أقل كفاءة من التي تربطهما معا.

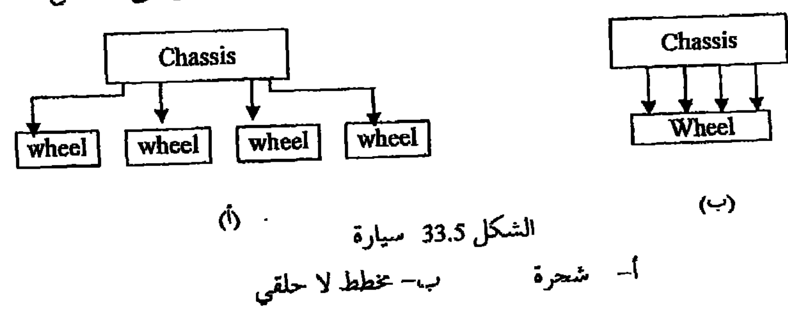
2- نظراً لكون القطع هي العناصر الأساسية في تطبيقات عالية المستوى لـذا ما تحتويه القطعة وكيف يتم تكوينها تعتبر قضايا حاسمة.

في منظومة GKS، تتألف القطع من كيانات أولية. يتم تمرير القطع إلى محطة العمل في فضاء إحداثيات الجهاز المعيارية (NDC). إذن شروط الرؤية متضمنة داخل القطعة، مع أنها بصورة غير مباشرة. أيضاً الصفات المميزة يتم ضمها بصورة غير مباشرة. على سبيل المثال، تكون الصفات الانفرادية ملازمة للكيانات الأولية وهكذا تكون في داخل القطعة. إن وجه النظر هذه تكون مشابهة إلى التي استخدمناه في مناقشتنا للملفات الملحقة (Metafiles) في الفصل الثالث.

نستطيع استنتاج ما يلي، تكون القطع (على الأقل مشابه إلى ما هو موجود في منظومة GKS) عبارة عن قوائم لكيانات أولية تفتقر إلى أية خواص هيكلية للنموذج. لهذا تكون القطع من هذا الشكل عبارة عن كيانات ساكنة (Static) أو ثابتة، تتيح إلى تغييرات محدودة فقط، مثل تغير الصفات المميزة للقطعة بعد تعريفها سابقاً. بما أن الاجتياز لنموذج التطبيق قد تم إنجازه أثناء تكوين القطعة، منظومة الرسومات لا تستطيع الاستفادة من هذا النموذج. أن تطبيق الصفات المميزة والتحويل المعياري كجزء من عملية تكوين القطعة لا تؤدي إلى طرق فعالة عند تغير المشاهد أو تعديل محتوى القطعة. إن هذا القصور يعتبر أحد المآخذ على منظومة GKS. إحدى الحلول هي إعادة تقييم أسلوب التعامل مع القطع والنمذجة.

2.9.5 مخططات لاحلقية إتجاهية (Directed Acyclic Graphs)

لنأخذ بنظر الاعتبار النموذج البسيط للسيارة الذي سبق شرحه. عند تمثيله كشجرة، نحصل على النموذج الهرمي البسيط ثنائي المستوى كما مبين على جهة اليسار من الشكل 33.5.



الشكل 33.5 سيارة

أ- شجرة ب- مخطط لا حلقي

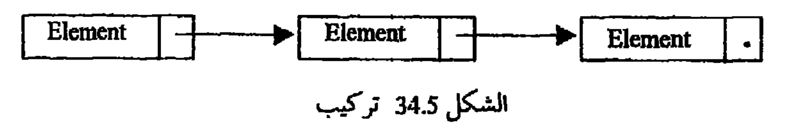
الرمز في المستوى العلوي يشير إلى الكيانات الأولية التي تصف فيه الهيكل، بينما الرمز المشار إليه في المستوى السفلي جميعها تشير إلى حالات مشاهدة لنفس الرمز وهي الدولاب. لنفترض أننا قمنا باحتواء الرموز للهيكل والدولاب في الرسم بدون التمييز بين محتويات العقدة ومحتويات الرمز. حيث نحصل على الرسم المبين على جهة اليمين من الشكل 33.5 الذي هو ليس بشجرة ولكن بالأحرى هو نوع من مخطط يعرف بـ "مخطط لا حلقي موجه" (غير دوري). في مخطط لا حلقي موجه، كل حافة لها اتجاه، لكن لا توجد فيه حلقات (Loops). الأشجار هي حالة خاصة من مخططات لا حلقية موجهة.

3.9.5 تراكيب (Structures)

نستطيع الآن أن نعطي مفهوم واسع للقطعة لكي تتيح لنا أن نصف مخططات لا حلقية. سنطلق على هذه الأنواع من البيانات بالتراكيب (Structures)، من أجل تميزها من القطع أولاً وأن تتوافق مع التسمية المستخدمة في منظومة PHIGS ثانياً. سنقوم بتبديل وحدات البناء الثلاث (العقد، المصفوفات والرموز) باثنين آخرين فقد هما: تراكيب وعناصر (Structures and Elements). والتركيب سيحل محل الشجرة. إذن، الدالة

traverse – structure (root)

ستؤدي إلى رسم التركيب المشار إليه بواسطة المؤشر root. التركيب يحتوي على عناصر. هذه العناصر تتضمن الكيانات الأولية التي تعرف رموزنا. أيضاً قد تحتوي على مصفوفات التحويل والصفات المميزة. من الممكن تصور العنصر ببساطة كرقم أو رمز تعريف لدالة مكتبية للرسومات وبيانات الدالة. قد يكون التركيب كما هو مصور في الشكل 34.5. كطريقة بسيطة لتكوين تركيب قد يكون من خلال الإجرائين التاليين:



الشكل 34.5 تركيب

new – structure (root)

new – element (structure, elemet)

الإجراء الأول يقوم بخلق مؤشر إلى تركيب جديد. وأما الإجراء الثاني يضيف عنصر إلى التركيب الجديد. لربما توصلت إلى اكتشاف أن تركيبنا كثيراً ما يشبه تركيب بيانات

أكثر بساطة من الشجرة وهو قائمة (List). أنت على حق. مع ذلك، علينا أيضا جعل التراكيب إلى حد ما عامه أو شاملة لوصف المخططات اللاحلقية الموجهة. بسبب تطابق عناصرنا مع إجراءات في منظومة الرسومات ، حيث يصبح نموذجنا بعض الشيء ديناميكياً أو حركياً. حيث لا تكون حاجة إلى تطبيق مصفوفات التحويل والصفات المميزة على الكيانات الأولية أثناء تكوين التركيب. نظراً لكونها مخزونة بصورة مباشرة داخل التركيب، لذا يمكن استخدامها حسب الضرورة أثناء اجتياز التركيب عند العرض. تعرف هذه العملية كاجتياز مقيد بالزمن (Traversal-Time Binding) . مثل هذه التراكيب تتيح كثيراً من أعمال منظومة الرسومات أن تنقل إلى محطة العمل الحقيقية. بالإضافة، أنها تؤدي إلى إمكانية أن يتم التغيير أو التنقيح، أما قبل عرضها أو بين العروض المتتالية.

نحصل على علاقات بين التراكيب بواسطة إدخال عنصر من نوع جديد، يدعى تركيب تنفيذ (Execute Structure)، الذي يشير إلى تركيب آخر. أثناء الاجتياز، هذا العنصر يشير إلى تركيب آخر الذي يتم اجتيازه (تنفيذه) قبل الاستمرار بتنفيذ تركيب الأب (الذي قام بالاستدعاء – Calling). الكيانات مثل مصفوفات التحويل والصفات المميزة بالإمكان تمريرها أو توارثها من تركيب الأب. قد لا تحتوي التراكيب على إشارات إلى نفسها أو إلى تراكيب أخرى التي تكون مرتبطة معها. أي بمعنى آخر، نحن لا نسمح بوجود حالة الدوران أو الحلقات (Cycles or Loops).

قد يكون تمرين بسيط في خلق تراكيب بإمكانها تمثيل أمثلتنا السابقة، التي استخدمت الأشجار. في تطبيق التراكيب، الصعوبة تكمن في عملية الاجتياز. هنا، علينا الاهتمام بعملية تنفيذ إجراءات الرسومات، بخلاف استخدامنا للقطع المشابهة في منظومة GKS، وذلك عند افتراض كون إجراءات الرسومات تم تنظيمها بطريقة بسيطة من خلال المصفوفات وجدول الرموز. الآن على عملية الاجتياز أداء التحويلات الضرورية والقيام بعرض الكيانات الأولية مع صفاتها الصحيحة. وهكذا إذا اعتبرنا أن العناصر تحتوي على ثلاثة حقول كما مبينة أدناه:

```
typedef struct
{
    int type; /* element type */
    record *data ; /*depends on element type */
```

```
        element *next:
    }     element;
```

فإنه يمكننا كتابة هيكل الإجراء كما يلي:

```
traverse – structure (root)
{
   element *pointer;
   pointer = root;
   while (pointor ! = NULL)
   {
      execute – element ( element . type , element . data);
      pointer + element. next
   }
}
```

حالياً تم إخفاء التعقيدات في الإجراء execute – element الذي يجب أن يكون قادراً على تنفيذ كل دالة ومن ضمنها تنفيذ التراكيب.

قد ترغب في محاولة تنفيذ منظومة نمذجة كهذه. أيضاً لربما ترغب الأخذ بنظر الاعتبار تصحيح بعض القصور في تراكيبنا البسيطة. على سبيل المثال ، الإجراء new-element لا يتيح لنا إضافة عناصر عند نقاط اختيارية في التركيب. إننا نحتاج مثل هذه المرونة إذا كنا نريد تعديل أو تنقيح هذه التراكيب. أيضاً إننا لم نتطرق إلى قضايا تتعلق بوجوب ا لتعامل مع المصفوفات (التي تعتبر عناصر مهيكلة) محلياً (التي تستخدم فقط ضمن التركيب التي تم فيها التعريف)، أو بصورة شاملة (ترحل إلى الأسفل لتشمل كل التراكيب المنفذة من قبل التركيب الذي تم فيه تعريفهم). سوف ننهي هذا الفصل بدراسة مقتضبة لمنظومة PHIGS وكيفية التعامل مع بعض هذه القضايا.

10.5 منظومة *PHIGS*

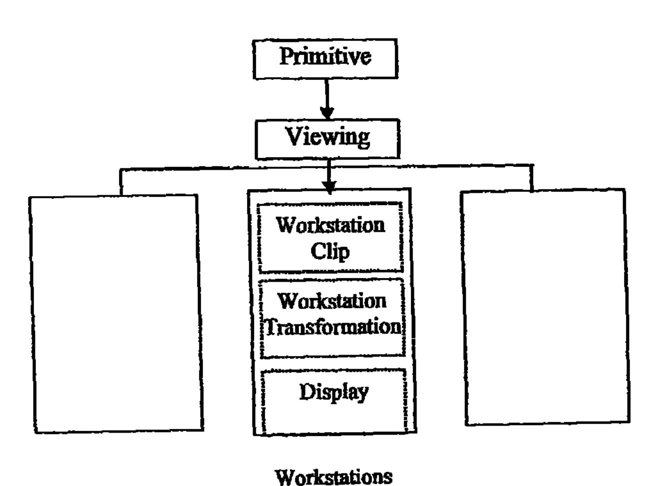
تكون النماذج الهرمية أساس لكثير من التطبيقات التي تستخدم الرسومات بالحاسوب. مع أن الإجراءات التي تم تطويرها لتناول التحويلات والنمذجة بسيطة، فنحن نفضل أن نبدأ مع منظومة رسومات تحتوي على إجراءات للنمذجة الهرمية. نحن لدينا مبررات للحاجة إلى مثل هذه البرمجيات لكي نذهب إلى خطوة أبعد من رغبتنا في تجنب

كتابة برامج إضافية. حيث تكمن رغبتنا في القيام باستخدام أقصى ما يمكن من قـــدرات المكونات المادية (والبرمجيات) لمحطات العمل الحديثة.

إن استحداث منظومة PHIGS كامتداد لما هو موجود من قياسيات الرسومات كـــان لحاجتنا في إضافة النمذجة لتلك المنظومات. مع أنه غير متناغم تماماً مع منظومـــة GK S، إلا أن الذين قاموا بتطويره حرصوا على جعله متناغماً بقدر الإمكان وبدون انتـــهاك أهـــداف تصميمه. سنقوم بالتعرف على اثنين من سماته المهمة وهي: قدراته في النمذجـــة وأسســـه المفاهيمية في المشاهدة لقاعدة بيانات رسوماتية (Graphical Databasc).

1.10.5 مشاهدة قاعدة بيانات (Viewing a Database)

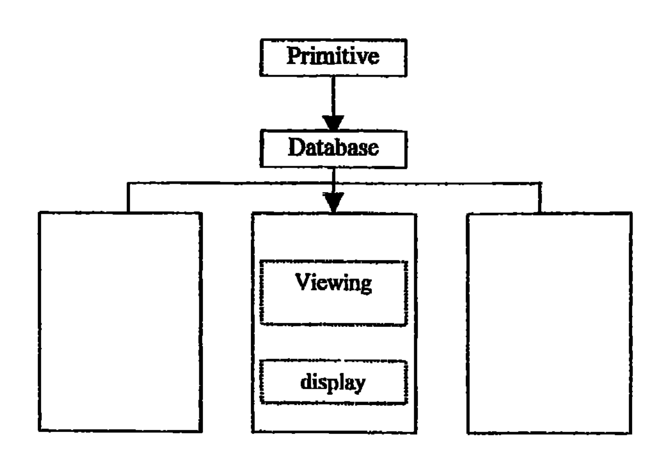
يوضح النموذج في منظومة GKS المبين في الشكل 35.5 أســـاس تدفـــق بيانـــات الكيانات الأولية ابتداءً من المواصفات إلى العرض.



الشكل 35.5 إنسيابية البيانات في منظومة GKS

يتم تطبيق التحويل الحالي للمشاهدة على جميع الكيانات الأولية، أثناء تكوين كـــل كيان أولي. بالرغم أن ، كل محطة عمل يمكنها رؤية جزء مختلف من فضاء NDC وذلـــك بأن يكون لها تحويل محطة عمل خاص بها، لكننا نمتلك قدرة محدودة فقـــط بالنســـبة إلى مختلف المشاهدين على محطات عمل حقيقية لعرض نفس البيانات بطرق مختلفة.

النموذج الأكثر شمولية، والذي يكون بالتأكيد مناسب لإجراء فيه عدد من المستفيدين قادرين على التعامل مع قاعدة بيانات كبيرة من خلال شبكة حاسبات كما مبين في الشكل 36.5.



Workstations 1

الشكل 36.5 نموذجة المنظومة PHIGS

الذي يستخدم في منظومة PHIGS. هنا ، يتم وضع الكيانات الأولية في قاعدة بيانات مركزية. كل مشاهد يمكنه الوصول إلى أياً من البيانات، ويتم تنفيذ عملية الرؤية بواسطة محطة عمل المشاهدة بصورة انفرادية. توضع التراكيب في قاعدة البيانات من قبل برامج المستفيد وقد يتم تنقيحها وحفظها لغرض الاستخدام لاحقاً. كما تم بحثه في البند السابق، تشمل عناصر التركيب على كيانات أولية، صفات مميزة وتحويلات. الصفات المميزة لا تكون مقيدة بالكيانات الأولية قبل عملية الاجتياز لغرض المشاهدة. هذا الإلحاق المتأخر (Late Binding) للصفات المميزة والقدرة على أن تكون لدينا محطات عمل مختلفة لها مشاهدات مختلفة للبيانات يعطي منظومة PHIGS صفة ديناميكية أكثر بعداً من منظومة GKS. بالإضافة، وكما هو معلوم أن المشاهدة هي وظيفة تنجزها محطة العمل، في حالات حيث تكون لدينا محطات عمل حقيقية متعددة تتقاسم قاعدة بيانات مشتركة، تكون منظومة PHIGS أكثر كفاءة بكثير من منظومة GKS.

2.10.5 البرمجة في منظومة PHIGS (Programming in PHIGS)

جميع برامجنا السابقة في منظومة GKS هي مطابقة فعلياً لتلك البرامج في منظومة PHIGS. الكيانات الأولية ومعظم الإجراءات هي نفسها. تبدأ أسماء الإجراءات مع الحرف 'p' بدلاً من 'g'. على سبيل المثال، يصبح إجراء متعدد الخطوط ppolyline. يتم إرسال تراكيب PHIGS إلى محطات العمل، كنتيجة واضحة جراء النموذج المبين في الشكل 36.5. سلسلة من إجراءات كهذه:

```
popen – struct (LINE);
pplyline (num-pts, points);
ptext (location, string;
pclose- struct( );
```

تقوم بتكوين عناصر في قاعدة البيانات ، لكن هذه العناصر ليست مرتبطة بمحطة عمل و لم تجري إن عملية رؤيا. عندما يتم إرسال التركيب إلى محطة عمل واحدة أو أكثر وبذلك بواسطة استدعاءات كالتي مبينة أدناه:

```
ppost- struct (wk-id, struct, priority);
```

حيث تتم المشاهدة والعرض. إن شروط الرؤية يتم تحديدها بأسلوب مشابهة إلى ما هو مستخدم في منظومة GKS، باستثناء ما يلي، حيث يجب ترقيم النوافذ وبوابات الرؤية برمز تعريف محطة العمل. يكون هذا المتطلب هو نتيجة اعتبار الرؤية وظيفة محطة العمل، لهذا يسمح لكل محطة عمل أن يكون لها مشاهدة مختلفة (أو مشاهدات) من البيانات. إذن، ستكون مهمة تحويل جميع برامجنا النموذجية إلى منظومة PHIGS بسيطة.

3.10.5 النمذجة مع منظومة PHIGS (Modeling with PHIGS)

كما رأينا في منظومة PHIGS التركيب هو أساس النمذجة بخلاف القطعة في منظومة GKS، التركيب يمكن أن يشير إلى تركيب آخر بواسطة عنصر التركيب للتنفيذ (Execute Structure Element). إن هذه الإضافة (إلى مبرمج التطبيق) تتيح لنا في تعريف نماذج هرمية بسهولة وبشكل متراص (Compactly). لنأخذ بنظر الاعتبار نموذجنا للإنسان الآلي. قد تبدوا التراكيب كما يلي:-

```
popen – struct (ROBOT);
```

```
        .
        pexecute – struct (BASE);
        .
        .
pclose – struct (  );
popen – struct (BASE);
        .
        .
        pexecute – struct (LOWER-ARM);
pclose- struct (  );
popen – struct ( LOWER- ARM);
        .
        .
        pexecute – struct ( UPPER- ARM);
        .
        .
pclose – struct ( ):
popen – struct (UPPER – ARM);
.
pclose – struct (  );
```

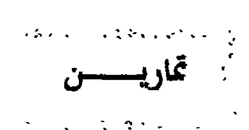
أو بالنسبة لشكل العصا:

```
popen – struct (BODY);
      pexecute – struct (TORSO);
      pexecute – struct (RIGHT – ARM);
      pexecute – struct (LEFT – ARM);
      pexecute – struct (RIGHT- LEG);
      pexecute – struct (LEFT- LEG);
pclose – struct (  );
popen – struct (LEFT- ARM);

         pexecute – struct (UPPER ARM);
pclose – struct  (  );
popen – struct (RIGHT – ARM);
         pexecute – struct (UPPER – ARM);
pclose – struct ( );
```

```
Popen-struct (RIGHT-ARM);
.
.
pclose – struct (  ):
.
.
```

إلى هذا البرنامج، ينبغي إضافة إمكانية تكوين وتعديل مصفوفات التحويل التي تقـوم بوضع العناصر المتصلة هرمياً حسب العلاقة بين الواحدة والأخرى. كما يتبين من المثـال، اثنان من التراكيب يمكنهما الإشارة إلى نفس الـتركيب في المسـتوى الأدنى(-UPPER ARM في مشكلة الجسم). هكذا يتم تكوين مخطط لا حلقي. إذا سبق تنفيذ الـتركيب مصفوفة تحويل صحيحة ستظهر لدينا حالتين مشاهدة للـتركيب UPPER – ARM في مواضع مختلفة مع أحجام واتجاهات مختلفة. أما بالنسبة للصفات المميزة للـتركيب يتـم توارثها من تركيب الأب الذي يعطي المبرمج مرونة أكبر.



تمارين

1.5 أعطي تعبير للتحويل المعياري (Normalization Transformation) أي (النقل من WC- إلى NDC)، بدلالة سلسلة من الانتقالات وتغيير الأبعاد. بين أن هذه السلسلة تعطي تمثيل مصفوفة صحيحة في إحداثيات متجانسة.

2.5 إحدى الطرق في تعيين تحويل تآلفي (Affine Transformation) وحيد يكون في تعريف مجموعة نقاط في كلا المواقع الأصلية ومواقعها الجديدة بعد أن يتم تحويلها. كم يكون عدد أزواج النقاط الفردية لتعريف تحويل.

3.5 بين أن القص (Shear) يمكن الحصول عليه من خلال التدويرات والانتقالات وتغيير الأبعاد.

4.5 برهن الخواص التالية لمصفوفات الانتقال، التدويل والقص:

$R^{-1}(\theta) = RT(\theta),$

$R(\theta + \phi) = R(\theta)\, R(\phi),$

$T(\Delta x_1 + \Delta x_2, \Delta y_1 + \Delta y_2) = T(\Delta x_1, \Delta y_1)\, T(\Delta x_2, \Delta y_2),$

$H_x^{-1}(\theta) = H_x(-\theta).$

5.5 بين إمكانية توليد جميع التحويلات التآلفية المحتملة بواسطة سلسلة مصفوفات التدوير وتغيير الأبعاد. هل من الممكن توليد أي تحويل تآلفي اختياري مع نوعين أساسين من التحويلات فقط.

6.5 حرك سيارة بواسطة بناء نموذج مناسب ومن ثم بواسطة تغيير المصفوفات في النموذج.

7.5 أعطي مثالاً لمشكلة تصميم حيث تكون النماذج الهرمية غير مناسبة أو غير كافية لبناء نموذج. ما هي الدلالات، إذا كانت موجودة، التي يحتويها مثالك لمنظومة الرسومات.

8.5 مربع طول ضلعه 2^N وحدة يمكن تقسيمه إلى بلاطات مربعة غير متداخلة، طول ضلعها 2^{N-1} وذلك بتنصيف المربع أفقياً وعمودياً معاً. وكل مربع ناتج يمكن تقسيمه اختيارياً بنفس الأسلوب. وكل من هذه البلاطات المربعة يمكن أن يخصص لها لون وشكل. أوصف تركيب بيانات لمثل هذا الشكل.

9.5 تستخدم الأشجار في علم الحاسبات في عدد من المجـالات ، مثـل البحـث (Search) والتبويب (Sorting) وفي تمثيل البيانات. استخدم الرسومات بالحاسوب لغرض تنفيـذ خوارزمية تعتمد على الشجرة، وذلك بواسطة توليد سلسلة من الصور التي تبين الشـجرة عند مراحل مختلفة للخوارزمية.

10.5 في كثير من منظومات الرسومات يتم استخدام مصفوفة التحويل الحالية (CTM) لبنـاء نموذج. الإجراءات المتوفرة للمستفيد تكون عادة كالمجموعة التالية:

```
accumulate – ctm ( matrix, ctm);
evaluate – ctm ( ctm);
trauslate – ctm (x, y, ctm):
rotate – ctm ( theta, ctm);
scale – ctm (sx , sy, ctm);
push-ctm ( ctm, stack);
pop – ctm( ctm, stack);
```

تقوم الإجراءات الخمسة الأولى السماح للمستفيد بتثبيت أو تغيـير المصوفـة CTM. عادة يتم الاحتفاظ بمصفوفة CTM باستخدام المكدس (Stack). صمـم واسـتخدم الاجراءات السبعة في تنفيذ نموذج مهيكل للشجرة (Tree – Structured Model).

11.5 نفذ منظومة نمذجة مهيكلة بالشجرة بدون اللجوء إلى اسـتخدام التكـرار المتداخـل (Recursion).

12.5 ما هي فوائد ومضار استخدام تقنيات التكرار المتداخل في رسومات الحاسوب، بالمقارنـة مع استخدام أسلوب أكثر إجرائيـة (Procedural Aaproach) كمـا في منظومـة PHIGS؟

المصطلحات المعربة

"A"

Application Programmer	مبرمج التطبيق
Arcade Game	لعبة القناطر
Animation	صور متحركة
Algorithms	خوارزميات
Attributes	صفات
Arrow Keys	مفاتيح ذات أسهم
Absolute Positioing	تحديد موقع مطلق
Arbitrary Implicit Equation	معادلة ضمنية اختيارية
Aspect Ratio	نسبة مربع أقصى
Allocating Memory	تخصص ذاكرة
Activation	تنشيط
Alignment	تراصف
Arbitrary	اختياري (كيفي)
Affine Transformation	تحويلات تآلفية
Accumulation Procedures	إجراءات تراكم
Animation The Model	حركة النموذج
Abstract Data Types	أنواع بيانات تجريدية
Amplifier	مضخمة
Alpha	أبجدي
Alphanumeric	أبجد عددي
Antialiasing	إزالة التشويه
Axonometric Projections	إسقاطات محورية
Ambient Light	الضوء المحيط
Approximating a Circle	تقريب الدائرة

"B"

Bit	رقم ثنائي، ثنائي، بت

Boards	ألواح إلكترونية، كارتات
Bounding Box	إطار تحديد
Bundled Attributes	صفات مرزومة
Batch Processor	معالج دفعات
Background Color	اللون الخلفي (الخلفية)
Brightness	النصوع
Back-Pointing Face	وجه تأشير خلفي
Blending Polynomials	عديدات الحدود التوليفية
Bezier Polyomials	عديدات حدود بزير
Bicubic Polynomials	عديدات الحدود ثنائية التكعيبية
Bezier Patches	رقع بزير
Boundary Representation	تمثيل الحدود

"C"

Cathod- Ray Tube (CRT)	أنبوبة الأشعة الكاثودية
Calligraphic	راسم خط
Chip	شريحة: رقيقة للدوائر الإلكترونية
Circuit-Layout Package	حزمة برامج لتخطيط دائرة كهربائية
Click On	أنقر أو طقطقة
Charts And Graphs	مخططات وخطوط بيانية
Computer-Generated Images	صور مولدة بالحاسوب
Computer-Graphics	رسومات بالحاسوب
Computer-Aided Design (CAD)	تصميم معزز بالحاسوب
Complex Pictures	صور مركبة
Complex Input Device	جهاز إدخال مركب
Commands	أوامر
Configurations	أشكال تكوينية
Cursor	المنـــزلقة
Control Function	دالة سيطرة (تحكم)
Clipping	تقليم

Current Position	الموقع الحالي
Cell Arrays	مصفوفات خلية (صفوف خلية)
Cross-Batch Pattern	متبللة عرضية الشكل
Control	سيطرة (تحكم)
Colour Index	دليل اللون
Choice	اختيار
Create Segment	تحديث قطعة
Current State	الحالة الجارية
Control Loop	دورة السيطرة
Contacting Transformation	تحويلات متسلسلة
Chassis	الشاصي (الهيكل)
Child	ابن
Carriage-Return (CR)	رمز عودة العربة
Compatible	تناغم
Coprocessor	معالج مساعد
Concurrent Processes	العمليات المتزامنة
Convex Polygons	مضلعات محدبة
Coherence	تلاحم
Cyan	أزرق داكن
Color Lookup Table	جدول بحث اللون
Classical Viewing	المشاهد التقليدية
Cast Rays	أشعة ساقطة
Cubic Polynomials	عديدات الحدود التكعيبية
Cubic Interpolating Polynomials	عديدات الحدود التكعيبية المستكملة
Constructive Solid Geometry	الهندسة المجسمة التركيبية
Continuity	الاتصال
Convex Hull	غلاف محدب

"D"

Data Structure	تراكيب البيانات

Data Tablet	لوحة بيانات
Display Processing Unit	وحدة معالجة العرض
Display Memory	ذاكرة العرض
Display File	ملف عرض
Display Processor	معالج عرض
Display of Information	عرض المعلومات
Displays	عارضات صورية
Design	تصميم
Deflection-plates	ألواح الإنحراف
Direct-Views Storage Tube (DVST)	أنبوبة التخزين للرؤية المباشرة
Device Driver	وحدة إدارة الجهاز
Default	بدائل افتراضية
Debugging	تشخيص أو كشف
Debugger	كاشف الأخطاء
Device Coordinate	إحداثيات الجهاز
Device Independent	استقلالية الجهاز
Data Record	قيد بيانات
Default Value	قيمة افتراضية
Data Bases	قواعد بيانات
Deactivate	أبطال فعالية
Detectability	الاكتشافية (قابلة للاكتشاف)
Dragging	سحب
Dials	مزوال
Directed Acyclic Graphs	مخططات لاحلقية اتجاهيه
Destination Pixel	عنصر صورة المقصد
Digital-To-Analogue Converter (DAC)	محول رقمي إلى تناظري
Direction Cosine	جيب تمام الاتجاه
Direction Vectors	متجهات الاتجاه
Dimetric View	مشهد مزدوج الأبعاد
Depth Sort	ترتيب العمق

Diffuse Reflection	الانعكاسات الإنتشارية
Degenerate	انحلال
Determinant	محدده
Dumb-Bell	————

"E"

Emulator	منافس، محاكي
Electronic-Mail	رسائل بريد الإلكترونية
Electronic Office	مكتب سكرتارية الإلكترونية
Entities	كيانات
Explicit Curves	منحنيات صريحة
Error Handling	تناول الأخطاء
Error File	ملف الأخطاء
Ellipse	قطع ناقص/ أهليج
Enumerated	قابلة للعدد
Editor	محرر
Echo	صدى، إظهار
Event	حدث
Event Input	مدخلات الحدث
Event Queue	طابور الإحداث
Event Driven Input	إدخال مدفوع بالحدث
Evaluation Procedures	إجراءات تقويم
Extrusion	بروز
Exclusire-Or	أو المقصوره

"F"

Flight-Simulation	محاكي الطيران
File Cabinet	خزانة ملفات
Filicker-Free	خالية من الارتعاش
Font	طاقم حروف النص

Fluid Flow	تدفق السوائل
Floating Point	فاصلة سائبة
Frame Buffer	مخزن انتقالي للصورة
File Descriptor	واصف ملف
Fill Area	مساحة ملء
Flag	إشارة إعلام
Foreground Colour	اللون الأمامي
Feedback	تغذية مرتدة
Flashing	إيماض
Fine Tuning	موالفه دقيقة
Fill	ملء
Fill Solid Areas	مساحات صماء
Filter	مرشحه
Frustrum	شكل ناقص
Front-Pointing Face	وجه تأشير أمامي
Forward Differences	الفروق للأمام

"G"

Graphic Library	مكتبة رسومات جاهزة
Graphical Input Device	جهاز إدخال بياني
Graphical Entities	كيانات أساسية للرسومات
Graphical Primitives	كيانات أولية للرسومات
Graphical Output Primitives	كيانات الإخراج للرسومات
Graphical Kernel System(GKS)	منظومة نواة الرسومات
Graphics Terminal	طرفية للرسومات
Graphics Turtle	رسومات السلحفاة
Graphics Toolkits	أدوات رسومات
Geometric Attributes	صفات هندسية
Graphical Object	شيء منظور بيانياً
Graphical Devices	أجهزة إدخال مرسومة بيانياً

Graphical Database	قاعدة بيانات رسوماتية

"H"

Hardware	المكونات المادية
Hardcopy	نسخة مطبوعة
Hidden-Surface Removal	حذف السطوح المخفية
Hierarchical Models	نماذج هرمية
Host Computer	حاسوب حاضن أو مضيف
Hardware Line Generator	مكونات مادية لتوليد الخط
Header Item	فقرات المقدمة أو الصدارة
Hollow	مجوف
High-Resolution	درجة وضوح عالية/ دقة عالية
Highlighting	إظهار الإجراء ذات الأهمية
Help Facilities	وسائل مساعدة
Homogeneous Coordinates	إحداثيات متجانسة
Harsh Images	صور خشنه
Hermite Polynomials	عديدات حدود هيرمت

"I"

Interactive	تفاعلي
Icon	شواخص/دلالات
Input	مدخلات/إدخال
Instruction	إيعاز/ تعليمه
Input Devices	أجهزة إدخال
International Standard Organization (ISO)	منظمة التقييس الدولية
Initialize	تمهيد
Interactive Applications	تطبيقات متفاعلة
Inquiry Function	دالة استعلامية
Index	دليل أو مؤشر
Implementation Issues	قضايا تنفيذية

Interrupt	مقاطعة
Interactive Modeling	نمذجة متفاعلة
Implicit Screen Regeneration	تجديد الشاشة الضمني
Inverse Transformations	تحويلات معكوسة
Instances	حالات مشاهدة
Incremental Algorithm	خوارزمية تزايدية
Intensity	شدة الإضاءة
Instancing Cube	حالة مشاهدة معكب
Isometric View	مشهد متساوي القياس
Inward-Pointing	داخلي التأشير
Interpolation	الاستكمال

"J"

Joystick	عصا التحكم

"K"

Keyboard	لوحة مفاتيح

"L"

Light Pen	قلم ضوئي
Label Of The Axes	علامات المحاور
Lower Case	الحروف الصغيرة(غير الاستهلالية)
Line Style	شكل الخط
Logical File	ملف منطقي
Laser Plotter	راسم بيانات ليزرية
Locator	محدد موقع
Language Binding	ترابط لغوي
Logical Workstation	محطة عمل منطقية
Logical Input Classes	الصفوف المنطقية للإدخال
Logging	تسجيل الأحداث

Light Buttons	أزرار ضوئية
Line-Preserving Transformations	تحويلات محافظة على الخط
Linked List	قائمة موصولة
Laser Plotter	راسم بياني ليزر

"M"

Medical Impaging	تصوير طبي
Molecular Biology	بايولوجية الجزئيات
Mouse	الفأر
Mailbox	صندوق بريد
Multidimensional Data Generated	بيانات مولدة متعددة الأبعاد
Menu Driven Interfaces	واجهات بينية مسيرة بقائمة
Mouse-Controlled	جهاز إدخال مسيطر بالفأر
Menu-Driven Painting Program	برنامج طلاء مدار بقائمة اختيارات
Mapping	تخطيط أو نقل
Monochromatic	أحادية اللون
Metafiles	ملفات ملحقة
Menu	قوائم اختيار
Multiprocessing	متعدد معالجات
Menu-Driven Interactive System	منظومة متفاعلة مسيرة بقائمة اختيارات
Manual	دليل يدوي
Measure	قياس
Modeling	نمذجة
Matrix Transposition	إبدال مصفوفه
Magenta	الأحمر المزرق

"N"

Network	شبكات
Normalized Device Coordinates	إحداثيات قياسية (معيارية) للجهاز
Normalization Tansformation	تحويل قياسي أو معياري

Converted by Tiff Combine - (no stamps are applied by registered version)

"O"

Orthographic Projections	اسقاطات متعامدة
Office-Automation	نظام مكننة المكاتب
Output	مخرجات، إخراج
Output Devices	أجهزة إخراج
Objects	أشياء منظورة
Operating System	منظومة التشغيل
One-To- One Correspondence	تناظر أحادي
Output Primitives	كيانات أولية للإخراج
Overhead	عمل إضافي
Outcodes	شفرات خارجية
Oblique Projecctions	إسقاطات مائله
Orthogonal Viewing	مشاهده عموديه
Object-Space	فضاء-الأشياء المنظوره
Outward-Pointing	خارجي التأشير
Opaque Surface	سطح معتم
Object Classes	فئات الأشياء المنظورة

"P"

Pointing-Device	جهاز تأشير
Perspective Drawing	رسومات منظورية
Plotter	راسم خطي
Picture-Elements-Pixels	عناصر الصورة
Photorealistic Images	صور واقعية الإضاءة
Processor	معالج
Picture Formation Processing	معالجة تكوين الصورة
Positional Information	معلومة موقعية
Plotting Program	برنامج راسم بياني
Procedures	إجراءات

Parallel Architecture	معمارية متوازية
Positioning	تحديد موقع
Parametric Representation	تمثيل معلمي
Plotting Procedure	إجراء راسم بياني
Picture Formation	تكوين الصورة
Primitive	كيان أولي
Painting	تصوير زيتي
Photograph	تصوير ضوئي
Projector	خط الإسقاط
Projection Plane	مستوى الإسقاط
Pen Plotter	راسم قلمي
Pseudo Code	عبارات مستعارة
Programming Paradigm	نموذج برمجي
Parameter	معلمية
Physical Output Devices	أجهزة إخراج حقيقية
Portability	التنقلية
Polygon Vertices	رؤوس مضلع
Polymarkes	متعدد العلامات
Polynomial Curves	منحنيات عديدة الحدود
Pick	التقاط
Programmer's Mode	نموذج المبرمج
Procedure Calls	نداءات الإجراءات
Physical Units	وحدات حقيقية
Physical Workstation	محطة عمل حقيقية
Polylines	متعدد الخطوط
Pattern	مخطط / شكل
Prompts	توجيهات/ حث
Priority	أسبقية
Potentiometer	مقياس فرق الجهد
Pick Identifier	رمز تعريف الإلتقاط

Pulling Down Menus	سحب قوائم الاختيار
Parent	الأب
Pipeline	خط تنفيذ العمليات
Pseudo-Display File	ملف عرض مستعار
Punctuation Marks	علامات الترقيم
Parallelepiped	متوازي سطوح
Pull-Down Menu	قائمة اختيار مسحوبه للأسفل
Painting Algorithm	خوارزمية طلاء
Prism	منشور
Perceptual Color	إدراك حسي للون
Projection	إسقاط
Perspective Viewing	مشاهدة منظوريه
Parallel Projection	إسقاط متوازي
Planar Polygons	مضلعات مستويه
Polyhedra	متعدد الوجوه
Perfect Reflector	عاكسه تامة
Projection Reference Coordinates	إحداثيات مرجعية الإسقاط
Parametric Form	الصيغة المعلميه
Polynomial Curves	منحنيات عديده الحدود
Parametric Surfaces	السطوح المعلميه

"G"

Queue	طابور
Quadrilateral and Triangular Meshes	الشبكات الرباعية والمثلثية

"R'

Realistic Images	صور واقعية
Real-Time	الوقت الحقيقي
Ramdom Scan	مسح عشوائي

Refresh	إنعاش، تنشيط
Raster Graphics	رسومات شبكية
Raster-Scan	مسح شبكي
Rasterization	تشابك
Raster System	منظومة شبكية
Realistic Scenes	مشاهد واقعية
Rendering	طلاء
Relative Positioing	تحديد موقع نسبي
Raster Terminal	محطة طرفية للمسح الشبكي
Ragged	مثلمة
Recompile	إعادة ترجمة البرنامج
Request	طلب
Request Mode	نمط الطلب
Request Locator	طلب تحديد موقع
Reliable	وثوق
Rubber Band Line	خط النطاق المطاطي
Relaxation	تراخ
Rotation	تدوير
Robot Arm	يد الإنسان الآلي
Reflection	انعكاس
Resistor	مقاومة
Recursive	تكرار متداخل / معاودة
Reentrant Clipping	تقليم مقفل (قابل إعادة الدخول إليه)
Real-Time Processor	معالج في الزمن الحقيقى
Right Parallelepiped	متوازي سطوح قائم
Ray Tracing	اقتفاء الشعاع
Ray Casting	إسقاط الأشعه
Realism	الواقعيه
Replicating	تكرار
Recursive Subdivision	تقسيمات فرعيه مكرره متداخله

Refection Model	نموذج الانعكاس

"S"

Software	برمجيات
System Programmer	مبرمج المنظومة
Simulation	محاكاة
Software Package	حزم برمجيات
Scientific Visualization	الرؤيا العلمية
Super Computers	حواسيب عملاقة
Settings	إعدادات، تهيئات
Solid-State Memory	ذاكرة ذات الحالة الصلبة
Scan-Conversion	تحويل مسحي
Solid Modelers	نماذج مجسمة
Sophisticated Image-Generation	منظومات متطورة لتوليد الصور
Standard Type Of Arithmetic	عمليات حسابية اعتيادية
Self-Contained	تام في ذاته
Solid Area	مساحة مجسمة
Subarea	مساحة جزئية
Self-Scaling Plotter	راسم بيانات ذاتي التدرج
Standard Logical Units	وحدات منطقية قياسية
Standard Orgonizations	منظمات قياسيات
Serial Interface	أداة بينية متتابعة
State Tables	جداول حالة
String	صف من الرموز
Stroke	شوط أو ضربة
Segmentation Funtions	دوال التجزئة
Segments	قطع
Solid	متصل، ممتلئ
Setting Default Values	إعداد قيم البدائل الافتراضية
State Of The System	حالة المنظومة

Scaling Factor	معامل تقييس
Shape-Layout Program	برنامج تخطيط الأشكال
Solid Fill Area	مساحة ملء كلياً
Sample	عينة
Status Return	الحالة المعادة
Stylus	إبرة تسجيل
Sliding Down	الانزلاق للأسفل
Slide Bars	أشرطة منزلقة
Scaling	تغير أبعاد
Shear	قص
Symbols	رموز
Structured	تراكيب
Subtree	شجرة فرعية
Stick Figure	شكل العصا
Software Clipper	برمجيات المقلمة
Segmentation	التجزئة
Stand-alone	القائم بذاته
Spline Curves	منحنيات الوصلات
Swapping	المبادله
Source Pixel	عنصر صوره المصدر
Singularities	انفراديه
Spectrum	طيف
Shades of Gray	ظلال رمادي
Subtractive Color	لون إسقاطي
Spatial Curve	منحنى فضائي
Shadow	أشعة الظل
Specular Reflection	إنعكاس مرآوي
Solid-Modeling System	منظومه النمذجه الصلده
Surface Revolntion	دوران السطوح
Spatial Resolution	دقه فضائيه

Smoothing Polynomials	عديدات حدود ناعمه
Scan Converting Polynomials	عديدات حدود التحويل المسحي
Solid Modeling	نمذجه الأشكال الصماء
Scalar Polynomial	عديده حدود عدديه
Shallow	مصلطحه

"T"

Terminal	طرفية
Terminal Based Software	برمجيات طرفية القاعدة
Transport	نقل
Turnkey Software	برمجيات جاهزة
Transformation	تحويلات
Time-Shared	مشاركة زمنية
Terminal Workstation	محطة عمل طرفية
Termination	انتهاء
Thin	رفيع
Terminator Item	فقرة إنتهاء
Tolerances	تفاوت مسموح
Trigger	أداة القدح، زناد
Thumb Wheels	دواليب إبهامية
Trackball	كرة الماء
Translations	انتقال
Traverse	اجتياز
Tree-Structured Hierarchical Model	نموذج هرمي لتراكيب الشجرة
Triads	مجموعات ثلاثيه
Trapezoid	شبه المنحرف
Tag	وسم
Trimetric View	مشهد ثلاثي الأبعاد
Tertrahedron	رباعي السطوح المثلثيه
Threshold	عتبَه، بدء،حد
Tours	حلقه المرساه

"U"

User Interface	واجهات بيئة المستفيد
User Program	برنامج المستفيد
Unit Normal	عمود وحدة
Unit Square	مربع وحدة
User Interaction	التفاعل مع المستفيد
Upper Case	الحروف الكبيرة (الاستهلالية)
User Aids	إعانات للمستفيد
Utility Function	داله خدميه
Unit Vectors	متجهات واحديه
Unit Cube	مكعب وحده
Unit Magnitude	قيمه واحديه
Unit Basis Vectors	متجهات قاعده واحديه
Unit Interval	وحده فتره

"V"

Very Large Scale Integration (VLSI)	دوائر متكاملة ذات الكثافة العالية
Virtual	افتراضي
Vector	متجه
Viewing Pyramid	هرم الرؤية
Viewer	مشاهد
Viewport	بوابة الرؤية
Valuator	مقيم أو مخمن
Viewing	الرؤية أو المشاهدة
Virtual Screen	شاشة افتراضية/ ظاهرية
Volumetric Representations	تمثيلات حجميه
Viewing Reference Coordinates	إحداثيات مرجعيه الرؤيه
Vanishing Points	نقاط التلاشي
View-Plane Normal	عمود مستوى المشاهده
View-Orientation Matrix	مصفوفه توجيه الرؤيه

"W"

Waste Paper Basket	سلة الأوراق المهملة
World Coordinate	إحداثيات كونية
Window	نافذة
Workstation Transformation	تحويل محطة عمل
World Window	نافذة كونية
Wire- Frame Image	صوره ذات التأطير السلكي
Wire-Frame Model	نموذج إطار سلكي

"Z"

Zooming	تزويم، تكبير بالتقريب
Z-Buffer	مخزن إنتقالي – z–

BIBLIOGRAPHY

[Abel81] Abelson, H. and A. diSessa, Turtle Graphics, MIT Press, Cambridge, MA, 1981.

[Akel88] Akeley, K. and T. Jermoluk, " High Performance Polygon Rendering". Computer Graphics, 22 (4), 239-246, 1988.

[Aman 87] Amantides, J., " Realism in Computer Graphics: A Survey", : IEEE Computer Graphics and Applications, 7 (1), 44-56, 1987.

[Bar87] Bartels, R. H., Beatty, J.C., and Barskey, B.A. An In troduction to Splines for use in Computer Graphics and Geometric Modeling, Morgan Kaufmann, Los Altos, CA, 1987.

[Bres63) Bresenham, J.E., "Algorithm for Computer Control of a Digital Plotter," IBM Systems Journal, 4 (1), pp. 25-30, 1965.

[Bres87] Bresenham, J.E., "Ambiguities in Incremental Line Rastering," IEEE Computer Graphics and Applications, 7 (5), pp. 31-43, 1987.

[Bono85] Bono, P.R., "A Survey of Graphics Standards and Their Role in Information Exchange," Computer, 18 (10), 63-75, 1985.

[Bow83] Bowyer, A. and J. Woodwark, A Programmer's Geometry, Butterworth, London, 1983.

[bown85] Brown, M.D., and M. Hech, Understanding PHIGS, Megatek, San Diego, CA, 1985.

[Carl78] Carlbom, I. and J. Paciorek, "Geometric Projection and Viewing Transformations," Computing Surveys, 1(4), 465-502, 1978.

[Cat75] Catmull, E., "A Hidden- Surface Algorithm with Antialiasing," Computer Graphics, 12(3), 6-11, 1975.

[Cat75b] Catmull, E., "Computer Display of Curved Surfaces". Proceedings of the IEEE Conference Computer Graphics, Pattern Recognition and Data Structures, Los Angeles, CA. P. 11, May 1975.

[CG] Computer Graphics, ACM Special Interest Group on Graphics (SIGGRAPH) Association for computing Machinery.

[CGM96] "Metafile for the Storage and Transfer of Picture Description Information," ISO/DP 8632 International Standards Organization, 1986.

[Clark 82] Clark, J.E., "The Geometry Engine: A VLSI Geometry System for Graphics," Computer Graphics, 10(3), 127-133, 1982.

[cook82] Cook, R.L. and K.E, Torrance, "A Reflectance Model for Computer Graphics, "ACM Transactions on Graphics, 1 (1), 7-24, 1982.

[corn70] Cornsweet, T.N., Visual Perception, Academic Press, New York, 1970.

[Crow77] Crow, F.C., "The Aliasing Problem in Computer Generated Images" Graphics and Image Processing, 20(11), 799-805, 1977.

[Crow81] Crow, F.C., "A Comparison of Antialiasing Techniques," IEEE Computer Graphics and Applications, 1 (1), 40-49, 1981.

[Dun 83] Dunlavey, M.R., "Efficient Polygon Filling Algorithms for Raster Displays", ACM Transactions on Graphics, 2 (4), 264-273, 1983.

[Ear84] Earnshaw, R.A., Fundamental Algorithms for Computer Graphics, Springer – Verlag, Berlin, 1985.

[End84] Enderle, G., K. Kansy, and G. Pfaff, Computer Graphics Programming: GKS-The Graphics Standard, SpringerVerlag, Berlin, 1984.

[Faux80] Faux, I.D., and M.J. Pratt, Computational Geometry for Design and Manufacturing, Halsted, Chichester, England, 1980.

[Foley82] Foley, J.D. and A. van Dam, Fundamentals of Interactive computer Graphics, Addison-Wesley, Reading, MA, 1982.

[GKS84] Graphical Kernel System, ISO 7492, International Standards Organization, 1985. (Also ANSI X3. 124-1985, American National Standards Institute).

[GKS86] Graphical Kernel System for Three Dimensions, ISO 8805, International Standards Organization, 1986.

[Gold83] Goldberg, A. and D. Robson, Smaltalk-80: The Language and Its Implementation , Addison-Wesley, Reading, MA, 1983.

[Goral84] Goral, C.M., K.E. Torrance. D.P. Greenberg, and B. Battaile, "Modeling the Interaction of Light Between Diffuse Surfaces," Computer Graphics (SIGGRAPH 84), 18(3), 213-222, 1984.

[Guib82] Guibas, L. and J. Stolfi, "A Language for Bitmap Manipulation," ACM Transactions on Graphics, 1 (3), 191-214, 1982.

[Gour71] Gourand, H., "Computer Display of Curved Surfaces," IEEE Transactions on Computers, C-20 (6) , pp. 623-628, 1971.

[GSPC79] Graphics Standards Planning Committee, "Status Report of the Computer Graphics Standards Planning Committee," Computer Graphics, 13 (3), 1979.

[Hall83] Hall, R. and D.P. Greenberg, "A Testbed for Realistic Image Synthesis," IEEE Computer Graphics and Applications, 3 (8), 1-20, 1983.

[Harn83] Harrington, S., Computer Graphics : A Programming Approach, Mc Graw-Hill, New Yourk, 1983.

[Mearn86] Hearn, D. and M.P Baker, Computer Graphics, Prentice Hall, Englewood Cliffs, NJ, 1986.

[Mend86] Handerson, L.M. Journey, and C. Osland, "The Computer Graphics Metafile," IEEE Computer Graphics and Applications, 6(8), 24-32, 1986.

[Hop83] Hopgood, F.R.A., D.A. Duce, J.A. Gallop, and D.C, Sutcliffe, Introduction to the Graphical Kernel System: GKS, Academic Press, London, 1983.

[IGES86] Initial Graphics Exchange Specifcation, Version 3.0, NBSIR 86-3359, National Bureau of Standards, Gaithersburg, MD, 1986.

[Ing81] Ingalls, D., "The Smalltalk Graphics Kernel," Byte, 6 (8), 1981.

[Joy88] Joy, K.I., C.W. Grant, N.L. Max, and L. Hatfield, Computer Graphics : Image Synthesis, Computer Society Press, Washington, DC, 1988.

[Judd75] Judd, D.B and G. Wyszecki, Color in Business, Science and Industry, Wiley, New York, 1975.

[Ker88] Kenigan, B.W. and D.M. Ritchie, The Programming Language, Prentice-Hall, Engelwood Cliffs, NJ, 1988.

[Krebs79] Krebs, M. and J. wolf, "Design Principles for the Use of Color in Displays," Proceedings of the Society for Information Display, 20,10-15, 1979.

[Liang84] Liang, Y. and B. Barsky, "A New Concept and Method for line Clipping". ACM Transactions on Graphics, 3 (1), 1-22, 1984.

[Lien87] Lien, A., M. Shantz and V. Pratt, "Adaptive forward Differencing for Rendering Curves and Surfaces", Computer Graphics, 21 (4), 119-128, 1987.

[Mach83] Machover, C., "An Updated Guide to Sources of Information about computer Graphics," IEEE Computer Graphics and Applications, 1983.

[Mag87] Mangnenat- Thalman, N. and D. Thalman, Image Synthesis, Springer- verlag, Berlin, 1987.

[Man87] Mantyla, M. , solid Modeling, Computer Science Press, Rockville, MD, 1987.

[Mort85] Mortenson, M., Geometric Modeling, Wiley, New yourk, 1985.

[Murch84] Murch, G., "Physiological Principles for the Effective use of Color" IEEE Computer Graphics and Applications, 4 (11), 49-54, 1984.

[Myers88] Myers, B.A., "A Taxonomy of Window Manager-User Interfaces," IEEE Computer Graphics and Applications, 8 (5), 65-84, 1988.

[New73] Newman. W.M. and R.F. Sproull, Principles of Interactive Computer Graphics, McGraw- Hill, New York, 1973.

[Paper81] Papert, S. , LOGO: A Language for Learning, Creative Computer Press, Middletown, NJ, 1981.

[Pav82] Pavlides, T., Algorithms for Graphics and Image Processing, Springer-Verlag, Berlin, 1982.

[PHIGS86] Programmer's Hierarchical Interactive Graphics System, ISO TC97 SC21/N819, International Standards Organization, June 1986.

[PHIGS88] PHIGS+ Committee, "PHIGS + Functional Description, Revision 3.0," Computer Graphics, 22 (3), 125-215, 1988.

[Pike84] Pike, R., L . Guibas, and D. Ingalls, "Bitmap Graphics," Computer Graphics, 18 (3), 135-160, 1984.

[Phong75] Phong, B.T., "Illumination for Computer Generated Scenes," Communications of The ACM, 18 (6), 311-317, 1975.

[Post85] PostScript Language Reference Manual, Adobe Systems Inc., Addison- Wesley, Reading, MA, 1985.

[Pratt78] Pratt, W.K., Digital Image Processing, Wiley, New York, 1978.

[Reis81] Reisenfeld, R.F., "Homogeneous Coordinates and Projective Planes in Computer Graphics ," IEEE Computer Graphics and Applications, 1 (1) , 50-56, 1981.

[Rog76] Rogers, D.F. and A. J. Alan, Mathematical Elements for Computer Graphics , Mc Graw- Hill , New York, 1976.

[Rog85] Rogers, D.F., Procedural Elements for computer Graphics, McGraw- Hill, New York, 1985.

[Rog87] Rogers, D.F. and D.A. Earnshaw (eds), Techniques for Computer Graphics, Springer- Verlag, Berlin, 1987.

[Sal87] Salmon, R. And M. Slater, Computer Graphics: Systems and Concepts, Addison-Wesley, Workingham, England, 1987.

[Sch83] Schachter, B.J. (ed), Computer Image Generation, Wiley, New York, 1983.

[Schei86] Scheifler, R.W. and J. Gettys, "The X window Systems", ACM Transactions on Graphics, 5 (2), 79-109, 1986.

[Schnei87] Schneiderman, B., Designing the User Interface: Strategies for Effective Human -Computer Interaction, Addison- Wesley,

Reading, MA, 1987.

[Smith78] Smith, A.R, "Color Gamut Transform Pairs," Computer Graphics (SIGGRAPH), 12(3), 12-19, 1978.

[Smith84] Smith, B. and J. Wellington, , "IGES: A Key Interface Specification for CAD/CAM System Integration," Proceedings of Computer Graphics 84, 548-555, 1984.

[Smith82] Smith, D.C., C. Irby, R. Kimball, B. Verplank, and E. Harslen, "Designing the Star Interface," Byte, April, 242-282, 1982.

[Spr68] Sproull, R.F. and I.E. Sutherland, "A Clipping Divider," 1968 Fall Joint Computer Conference, 765-775, Thompson Books, Washington D.C. 1968.

[Spr85] Sproull, R.F. , W.R. Sutherland, and M.K. Ulner, Device Independent Graphics, McCraw – Hill, New York, 1985.

[Suth63] Sutherland, I.E., "SKETCHPAD: A Man-Machine Graphical Communication System" AFIRS Spring Joint Computer Conference, 329-346, Spartan Books, Baltimore, MD, 1963.

[Suth74] Sutherland, I.E. and G.W. Hodgeman, "Reentrant Polygon Clipping," Communications of the ACM, 17 (1), 32-42, 1974.

[Suth74b] Sutherland, I.E., R.F. Sproull, and R.A. Schumacker, "A Characterization of Ten Hidden – Surface Algorithms," Computer Surveys, 6 (1), 1-55, 1974.

[Tufte83] Tufte, E.R., The Visual Display of Quantitative Information, Graphics Press, Chesire, CN, 1983.

[Whit80] Whitted, T., "An Improved Illumination Model for Shaded Display," Communications of ACM, 23 (6) , 343-348, 1980.

[Wysz82] Wyszecki, G. and W.S. Stile, Color Science, Wiley, New York, 1982.

المحتويات

الجزء الثاني

الفصل السادس
التنفيذ

الفصل السابع
الرسومات ذات المسح الشبكي

الفصل الثامن
رسومات ثلاثية الأبعاد

مقدمة

الفصل العاشر
المنحنيات والسطوح





EDWARD ANGEL
University of New Mexico

COMPUTER GRAPHICS

Translated by :
Dr. N. A. Al-Sultany & Dr. B. Al-Hashimy

◆◆◆◆◆◆◆◆◆◆◆

لقد أصبحت رسومات الحاسوب ذات أهمّية قصوى للطلبة على اختلاف مستوياتهم ومجالات تخصصاتهم ، وللباحثين في مجال الحاسوب ؛ إذ إنّ التوسّع الذي طرأ في هذا المجال قد شمل ثروة جديدة ومثيرة من التطبيقات الّتي تبدأ من توليد صور واقعية الإضاءة إلى تصميم وسائل برمجيّات جاهزة .

إنّ الهدف من هذا الكتاب هو استخدامه في تدريس مادّة رسومات الحاسوب لطلبة علم وهندسة الحاسوب وتكنولوجيا المعلومات ، حيث إنّ الكتاب يغطي الكثير من المواضيع التي من شأنها الإيفاء باحتياجات الباحثين والطلبة .

لقد استخدم في هذا الكتاب الأسلوب الهرمي من الأعلى إلى الأسفل كي يتسنى للطلبة العمل على مشاريع تستخدم فيها منظومة نواة رسومات الحاسوب (GKS) أو أي منظومة أخرى باستخدام لغة C ، لما تتمتع به هذه اللغة من قدرة كبيرة في مجالات التصميم .

